



Universidad  
Continental

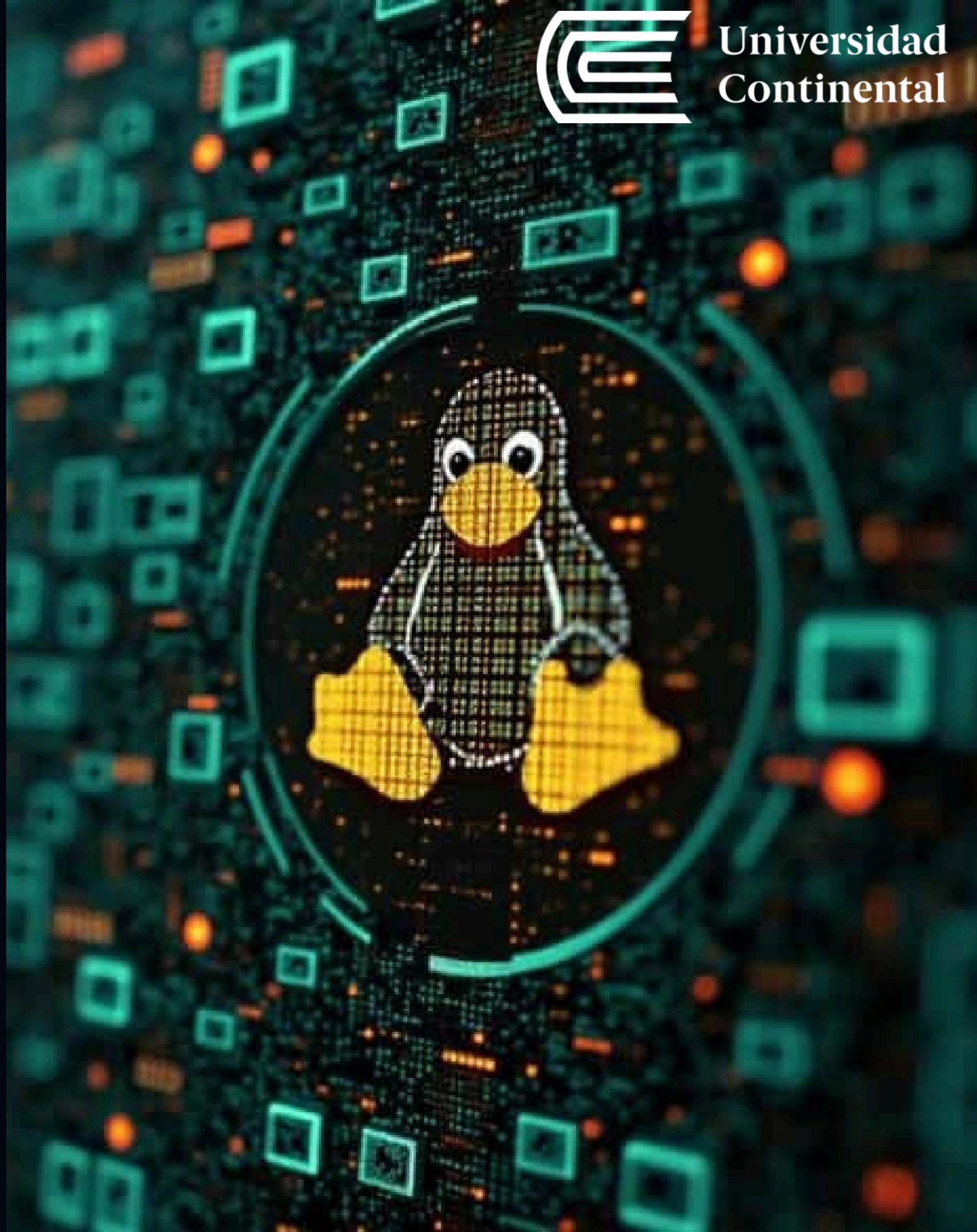
# PINGUEEXIT: SISTEMA OPERATIVO PERSONALIZADO PARA SALIDA SEGURA DE SESIÓN

Sistemas Operativos

Docente: Ing. Amir Fernando Mamdouh  
Mehrez Garcia

Grupo: 'G'

NRC: 17207





# INTEGRANTES



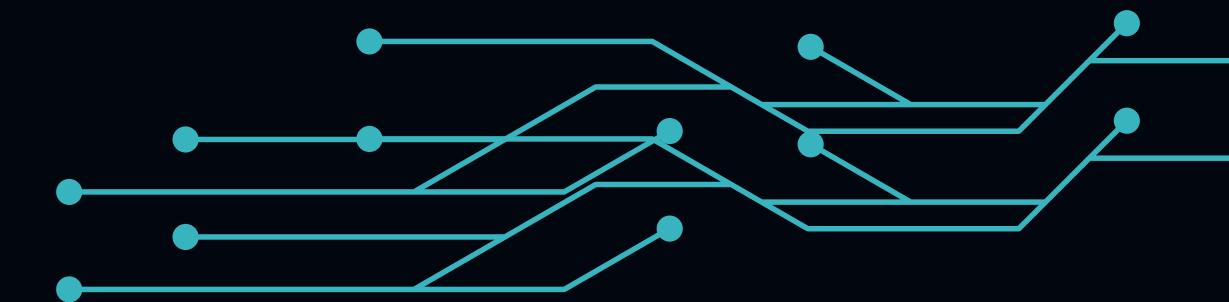
1. ZAVALA HUACARPUMA JUAN ALDAIR (100%)
2. GALINDO PEÑA ANYELA KELLY (100%)
3. CHALLCO CCOHUANQUI SAMIR JUNIOR (100%)
4. CORDOVA SUCAPUCA JODI (100%)
5. EMERSON MAMANI CAYAVILLCA (100%)



# 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.



**Evolution of Attack**

# ÍNDICE

1. Introducción
2. Planteamiento del problema
3. Objetivos
4. Marco teórico
5. Metodología
6. Desarrollo técnico
7. Resultados
8. Conclusiones
9. Anexos / Referencias

# INTRODUCCIÓN

El presente proyecto de investigación plantea el desarrollo y evaluación de la herramienta **IPPEExit**, 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.



Virtual Machine Wizard

ware  
ORKSTATION™

# 17

Welcome to the New Virtual Machine Wizard

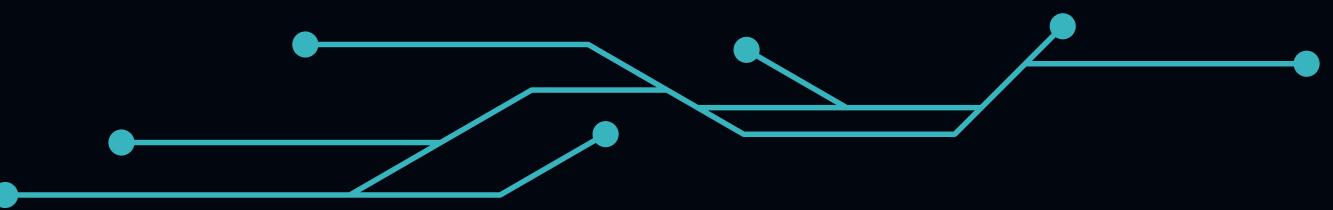
What type of configuration do you want?

Typical (recommended)  
Create a Workstation 17.5 or later virtual machine in a few easy steps.

Custom (advanced)  
Create a virtual machine with advanced options, such as a SCSI controller type, virtual disk type and compatibility with older VMware products.

Help < Back Next > Cancel





# PLANTEAMIENTO DEL PROBLEMA

Muchos usuarios de sistemas Linux enfrentan procesos poco intuitivos al cerrar sesión, especialmente cuando no tienen experiencia técnica. Esto puede llevar a errores o pérdidas de datos. Por eso, propusimos una solución con interfaz visual, clara y funcional.

En la actualidad, muchas redes locales domésticas o institucionales carecen de herramientas accesibles para su monitoreo y protección. La mayoría de usuarios no posee conocimientos técnicos avanzados en redes, lo que los deja vulnerables ante accesos no autorizados, ataques internos o intrusiones silenciosas.

Aunque existen soluciones profesionales para la gestión de redes, estas suelen ser complejas, costosas o poco intuitivas, y no están pensadas para un entorno educativo o doméstico. Además, herramientas capaces de detectar o expulsar dispositivos maliciosos generalmente requieren dominio de comandos o configuraciones avanzadas.

**En este contexto, se plantea el desarrollo de IPEExit, una aplicación que permita a cualquier usuario:**



**Visualizar qué dispositivos están conectados a su red.**



**Detectar intrusos en tiempo real.**



**Actuar frente a amenazas con técnicas controladas como el envío de paquetes ARP.**



# OBJETIVOS

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.

## GENERAL:

Desarrollar una aplicación de escritorio denominada **IPPEExit**, 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.

## ESPECÍFICOS:



Implementar interfaces gráficas intuitivas usando GTK.

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

Validar el correcto funcionamiento del software en un entorno de red controlado, midiendo su capacidad de detección y efectividad de expulsión.

# MARCO TEORICO

## BASES TEORICAS

En esta sección investigamos conceptos como:

- Sistemas operativos Manjaro basado en Linux.
- Interfaces gráficas (GTK).
- Biblioteca Scapy
- Gestión de IP en un entorno GUI.
- ARP (Address Resolution Protocol)
- Ética en el uso de herramientas de análisis de red



# METODOLOGÍA

IPPEExit fue desarrollado usando una metodología experimental y aplicada, con enfoque incremental. Esto permitió un proceso iterativo donde el programa se construyó y mejoró gradualmente, utilizando herramientas de desarrollo en Python.

## 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

Se analizaron escenarios comunes de intrusión en redes locales y se eligió ARP spoofing como técnica de detección e intervención.

## Implementación en Python:

Se implementó Scapy para enviar paquetes ARP personalizados, Tkinter para una interfaz gráfica interactiva y threading para mantener la ejecución continua sin bloquear la interfaz.

## Validación funcional:

Se realizaron pruebas locales con paquetes ARP maliciosos, observando efectos en la conexión y la restauración parcial de la red.

## Consideraciones éticas

Se destacó su uso exclusivo con fines educativos y pruebas en entornos controlados y autorizados.

# DESARROLLO TÉCNICO

Durante el desarrollo:

## Prerrequisitos y requisitos del sistema

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



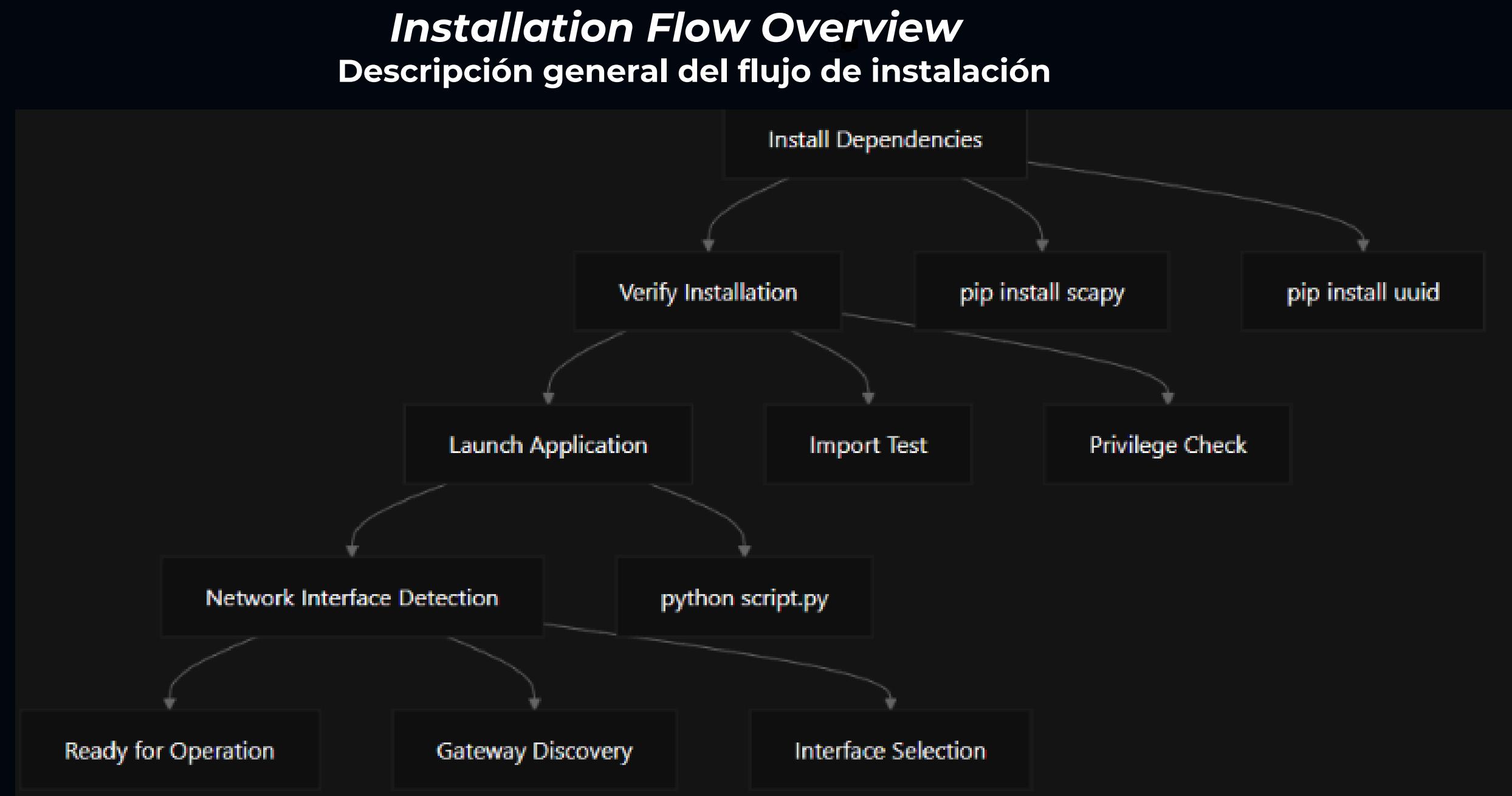
| Requerimientos    | Especificaciones                | Notas  |
|-------------------|---------------------------------|--|
| Sistema operativo | Linux, macOS, Windows.          | Root/Administrator privilegios necesarios para la manipulación de paquetes.    |
| Python Version    | Python 3.6+                     | 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. |

# ALGORITMO

## REQUERIMIENTOS PARA EL FUNCIONAMIENTO DEL ALGORITMO

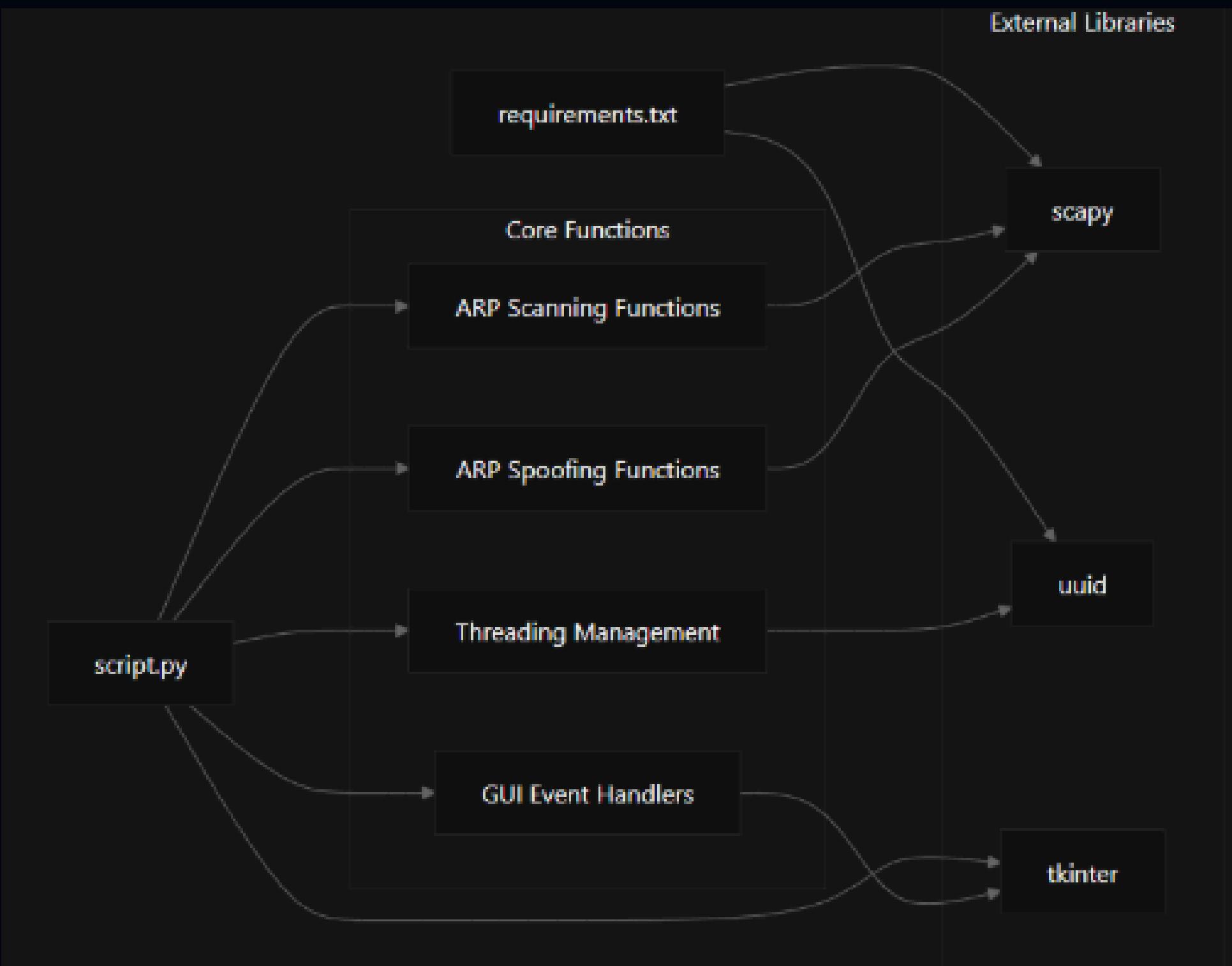
### Durante el desarrollo:

Es 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.



# ARQUITECTURA DE DEPENDENCIA

Es un diagrama de dependencias y estructura de un proyecto de aplicación de red desarrollado en Python, utilizando varias bibliotecas externas.



## Configuración de Privilegios

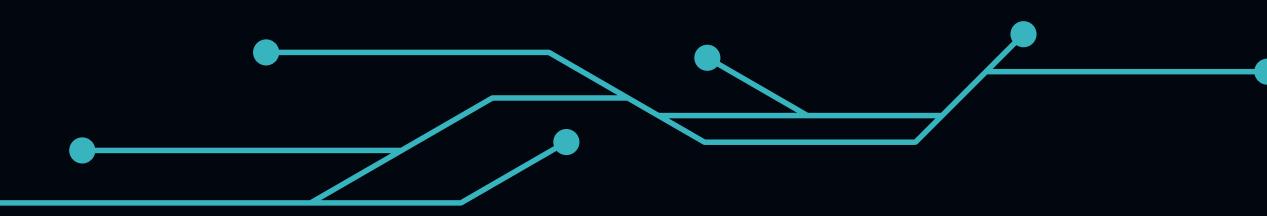
La manipulación de los paquetes de red requiere privilegios elevados, configuramos nuestro sistema Manjaro Linux.

### 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  | <code>python -c "import scapy.all"</code> | 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  | <code>arp -a</code> (línea de comando)    | Muestra todas las entradas ARP actuales. |

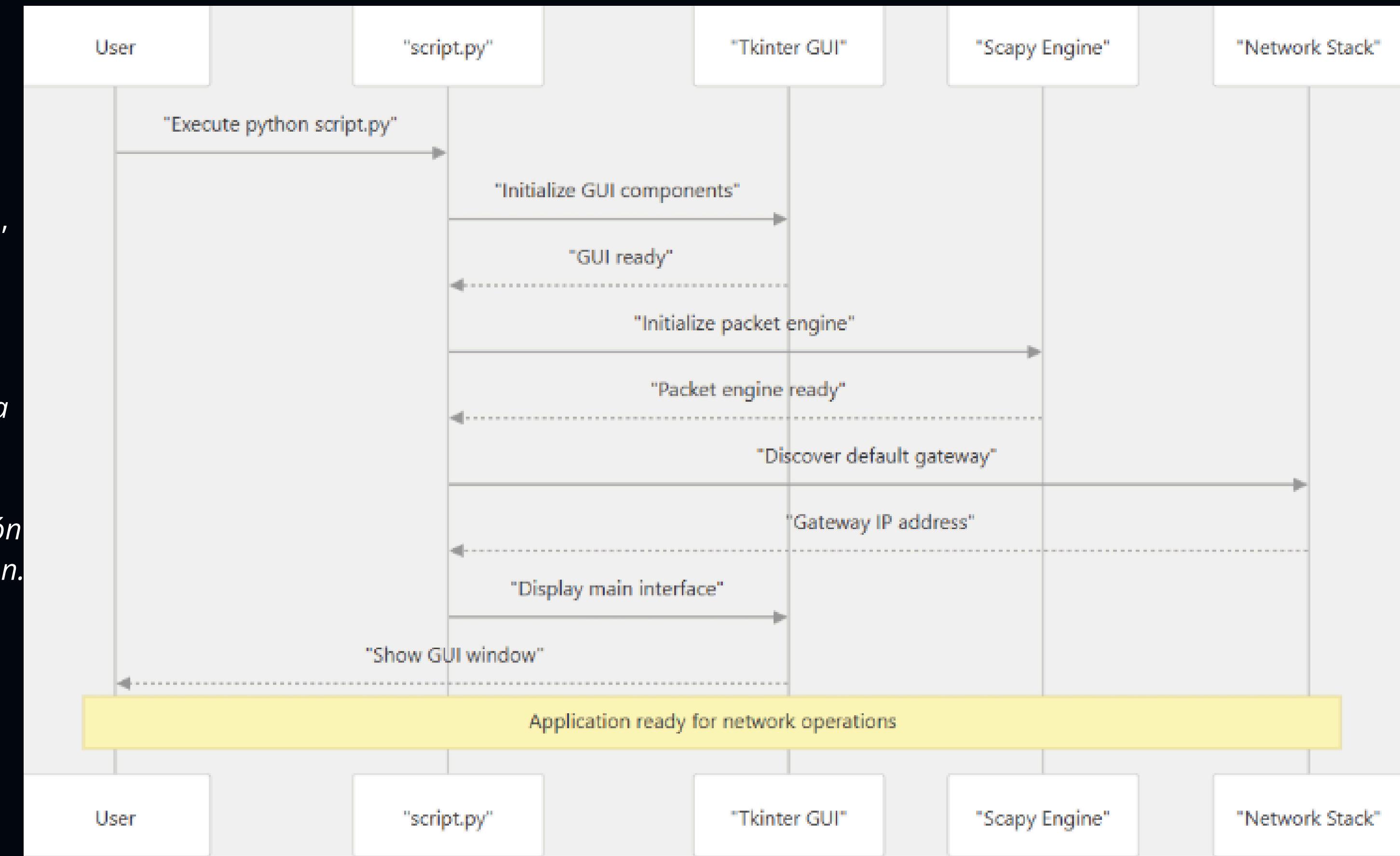
## Configuración del Entorno de Red

IPPExit 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.

# *Motor de paquetes basado en Scapy*

- 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:
  - *Lo cual muestra un diagrama de flujo que describe el proceso de inicialización y preparación de una aplicación de red desarrollada en Python.*



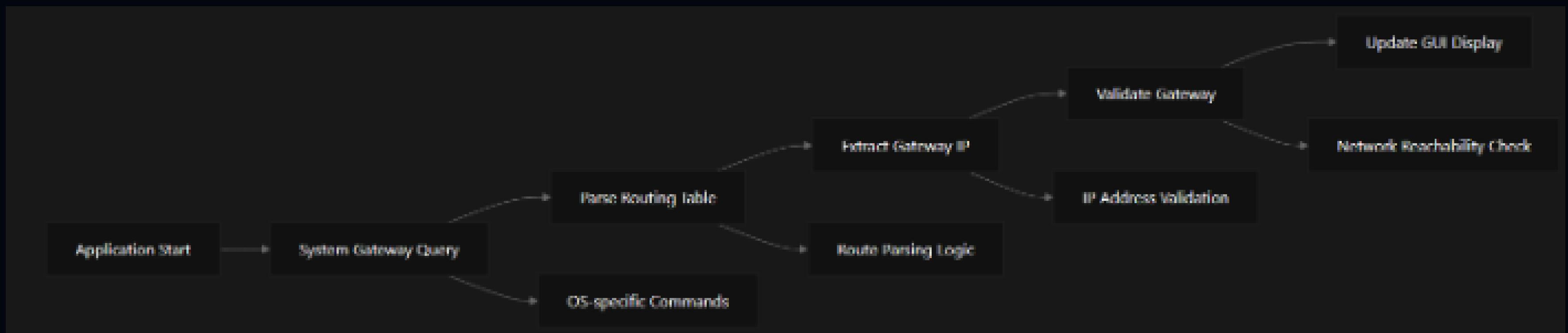
## Secuencia de prueba de Verificación

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.

## Proceso de detección de Puerta de Enlace

*Se muestra la aplicación que descubre automáticamente la configuración de la red utilizando utilidades de red a nivel del sistema.*



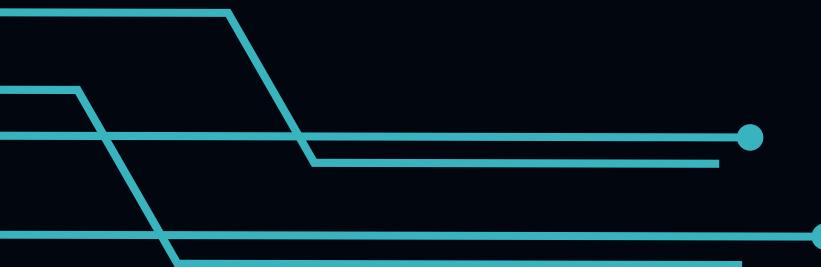
# ARQUITECTURA DE FLUJO DE DATOS

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.

*Se visualiza 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.*

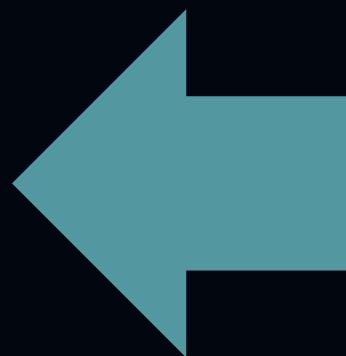
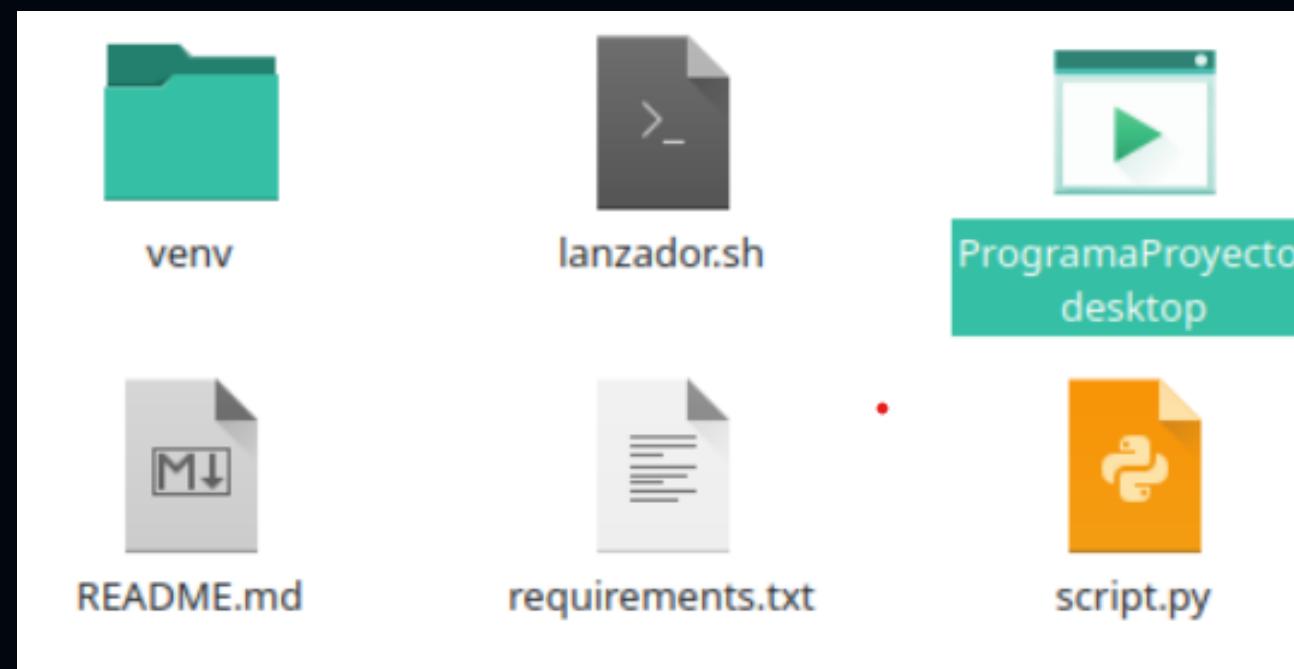


# ALGORITMO EN EJECUCIÓN

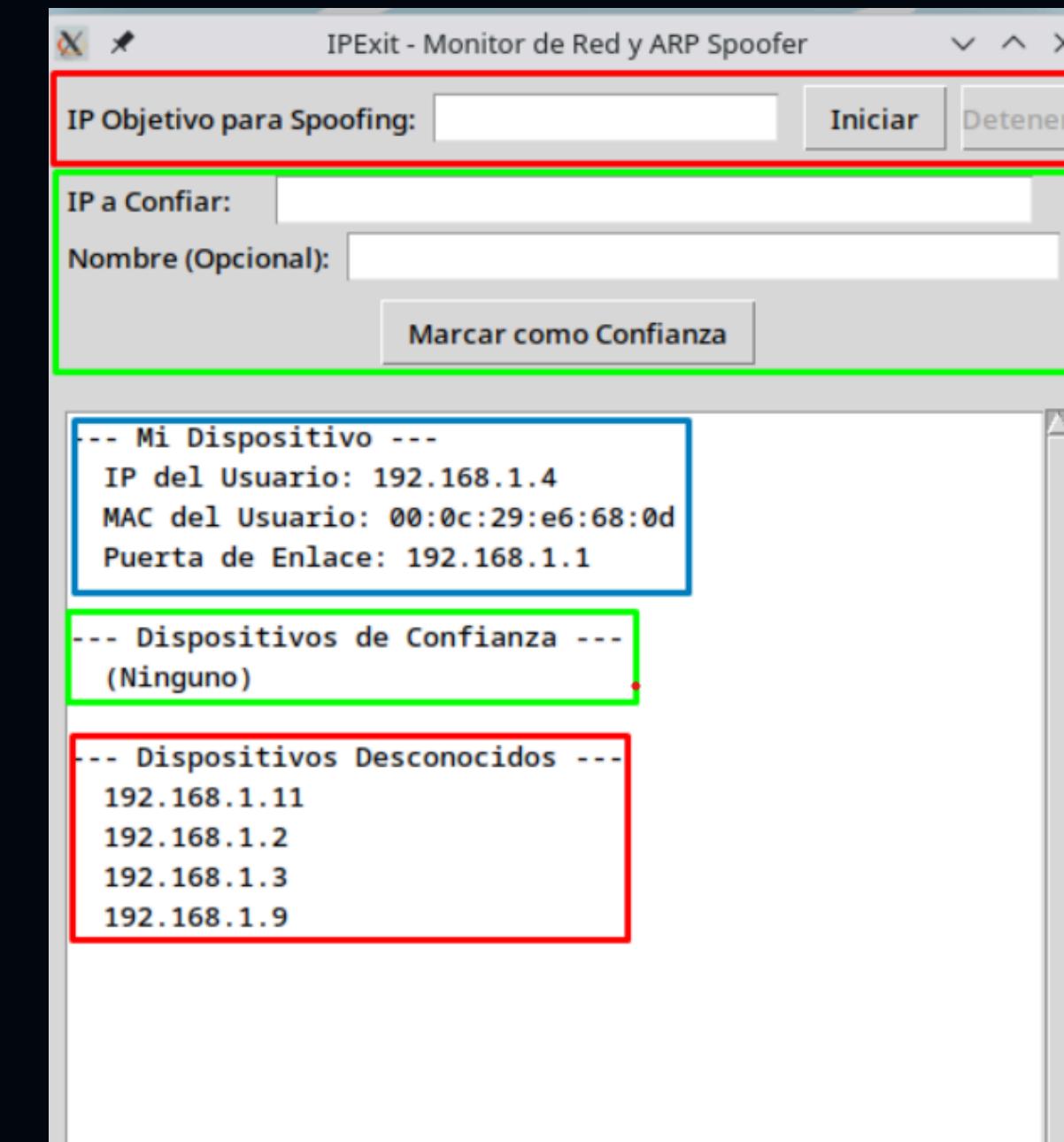


## COMPLEMENTOS

### ESTRUCTURA DE FICHEROS DEL PROYECTO IP EXIT

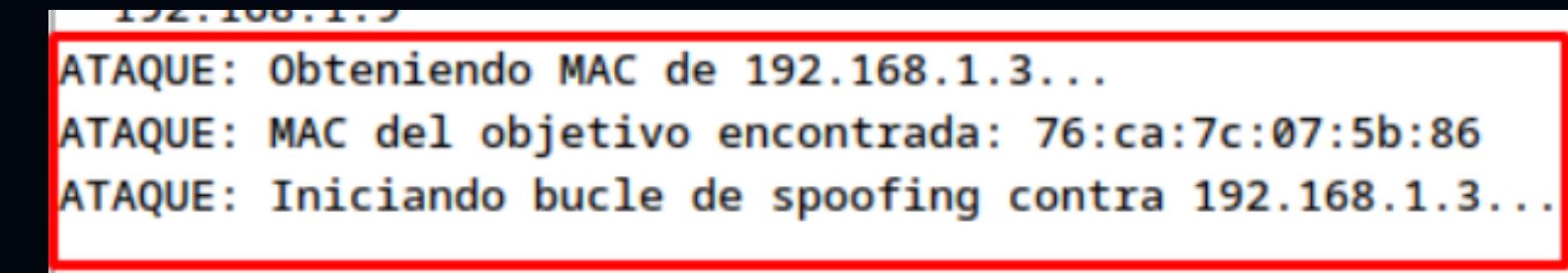


### INTERFACIAS DE SOFTWARE



# CASO 1 INICIO

# CASOS DE USO



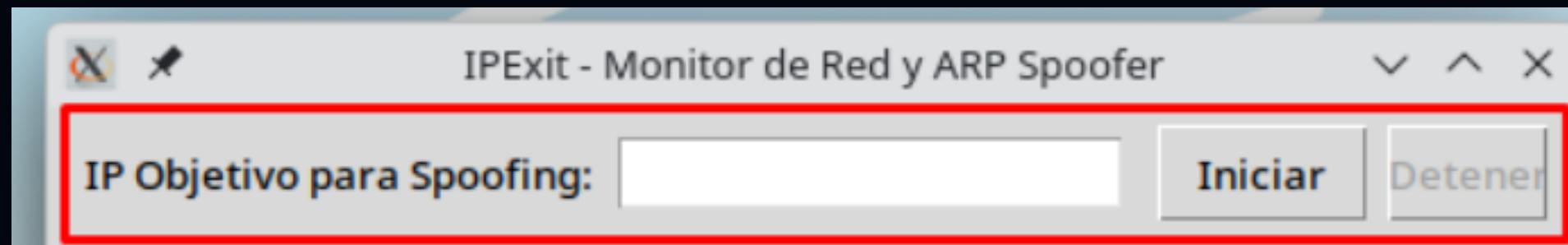
```
"Se ejecuta al pulsar "Iniciar". Valida la IP objetivo, obtiene su MAC y
lanza el hilo de spoofing."
if self.ataque_en_curso: return
ip_objetivo = self.entrada_ip.get()
if not ip_objetivo:
    messagebox.showwarning("Entrada inválida", "Por favor, ingrese una IP objetivo.")
    return
self.log_ataque(f"Obteniendo 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 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.

## CASO 2 STOP

# CASOS DE USO



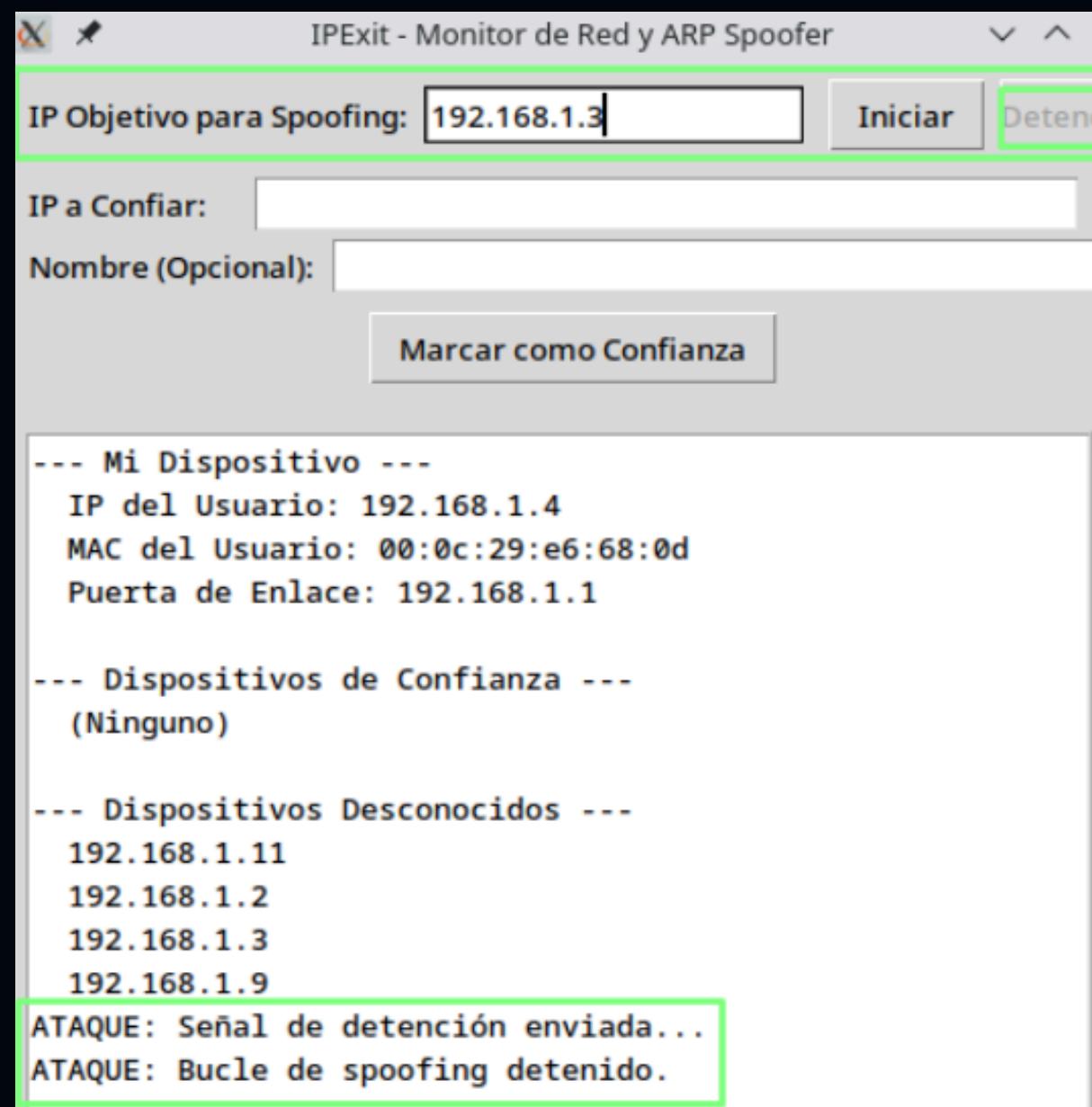
```
def detener_ataque(self):
    "Se ejecuta al pulsar \"Detener\". Para el bucle de spoofing y llama a la
    función para restaurar la red."
    if not self.ataque_en_curso: return
    self.log_ataque("Señal de detención enviada...")
    self.ataque_en_curso = False
    if self.hilo_ataque: self.hilo_ataque.join(timeout=3)
    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)
```

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.

# CASOS DE USO

## CASO 3 DETECCIÓN AUTOMÁTICA DE PUERTA DE ENLACE



```
# 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.
```

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:

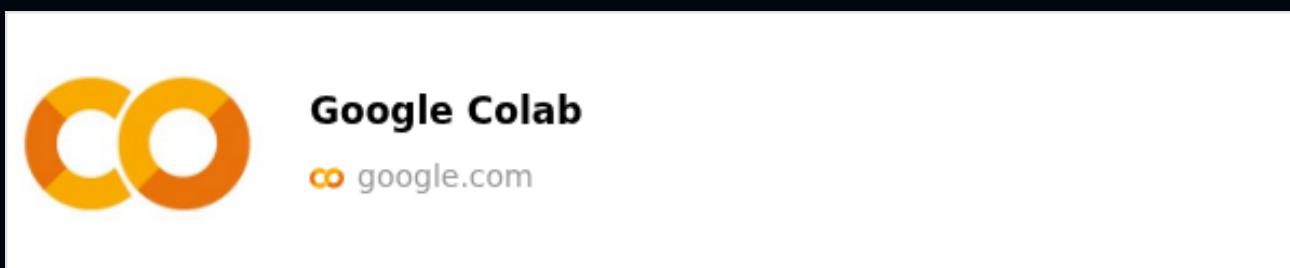
# CASOS

# TÉCNICO

- Los tres casos describen las funciones principales de una herramienta de ataque ARP: 1. Iniciar el ataque: Comienza el proceso de engañar a los dispositivos de red para redirigir su tráfico al atacante. 2. Detener el ataque: Finaliza el ataque y restaura el tráfico de red a su estado normal. 3. Detectar la puerta de enlace: Encuentra automáticamente la dirección IP del router, necesaria para el engaño.



En resumen, los tres casos ilustran el ciclo de vida completo de un ataque ARP usando la aplicación: iniciar, detener y la configuración necesaria para llevar a cabo el ataque. Es importante enfatizar que esta aplicación tiene el potencial de ser utilizada con fines maliciosos y su uso debe ser responsable y ético.



# 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](http://samples.jblearning.com)
- LEY DE DELITOS INFORMÁTICOS. Ley N.º 30096. [Gob.pe](https://www2.congreso.gob.pe). <https://www2.congreso.gob.pe>
- Manjaro Linux. (s.f.). *Manjaro Linux Empowering People and Organizations*. <https://manjaro.org>



**¡GRACIAS!**

