Analizador de paquetes e interfaces de redes con PyShark y Streamlit

Escalante Gaston Alejandro, Leopoldo Rios, Rodrigo Zalazar[1]

[1] Dpto. Informática, Fac. de Ciencias Exactas Naturales y Agrimensura, Universidad Nacional del Nordeste, Corrientes
gaston411@hotmail.com, ljr@comunidad.unne.edu.ar,
rodrigo.zalazar@comunidad.unne.edu.ar

Abstract. En el constante avance de las tecnologías de la información, el monitoreo efectivo de las redes informáticas se vuelve crucial. Esta tarea, inherentemente compleja, se ve agravada por la ausencia de herramientas especializadas que ofrecen una visualización intuitiva, dificultando la toma de decisiones de manera ágil.

Este trabajo de investigación surge como respuesta a la necesidad imperante de abordar este desafío. Presentamos una alternativa potente y accesible: una herramienta de código abierto con una interfaz intuitiva que busca simplificar el monitoreo de infraestructuras de redes informáticas.

Ubicado en el contexto de la asignatura de Redes de Datos, cuarto año de la Carrera de Licenciatura en Sistemas de Información, este proyecto no solo busca desarrollar e implementar esta solución, sino también evaluar su eficacia como una herramienta integral de monitoreo.

A lo largo de este documento, exploramos las complejidades inherentes al monitoreo de redes, discutiremos la elección y aplicación de bibliotecas de código abierto, y analizaremos cómo esta iniciativa contribuye al avance en la gestión de redes informáticas.

Keywords: PyShark, Streamlit, Wireshark, Monitoreo de redes, Análisis de Paquetes.

1 Introducción

La creciente complejidad de las redes informáticas en la actualidad ha destacado la necesidad imperante de contar con una herramienta integral que facilite la gestión y el control efectivo de estos entornos. En este contexto, se presenta la ineludible exigencia de soluciones que no sólo aborden los desafíos inherentes a la administración de redes, sino que también proporcionen una perspectiva sostenible y adaptable. En respuesta a esta demanda, se explora la viabilidad y las ventajas de las soluciones de código abierto, reconociendo su papel fundamental en la optimización de los procesos de monitoreo y gestión.

La elección de una solución de código abierto no solo responde a la necesidad de eficiencia y flexibilidad, sino que también refleja la filosofía de transparencia y colaboración que impulsa el desarrollo de herramientas en la comunidad informática. El acceso abierto al código fuente no solo promueve la personalización según las necesidades específicas, sino que también fomenta la innovación continua a medida que los usuarios contribuyen y mejoran la solución de manera colectiva.

Por otro lado, soluciones privadas como SolarWinds Network Performance Monitor (NPM) y Cisco Stealthwatch destacan por su enfoque especializado en el análisis de tráfico y la

seguridad de la red[1]. A pesar de estas soluciones privadas de renombre, es crucial reconocer la importancia de incorporar herramientas de código abierto en el arsenal de gestión de redes. Estas soluciones destacan por su flexibilidad, bajo costo y capacidad para adaptarse a diversas necesidades.

Continuando con nuestra exploración, nos sumergimos en una solución integral centrada en el monitoreo de paquetes de red: "Analizador de redes con PyShark y Streamlit". Esta propuesta innovadora aprovecha herramientas clave como Wireshark y Pyshark, integrando las ventajas de soluciones de código abierto en el análisis de tráfico de red.

Wireshark, reconocido como una herramienta de análisis de tráfico de red de código abierto, se erige como el pilar fundamental de esta solución, ofreciendo una visión profunda del flujo de datos en la red[2]. Complementando esta capacidad, Pyshark, wrapper de Python para tshark, que permite el análisis de paquetes de Python utilizando disectores Wireshark[3]. Esta sinergia proporciona no solo flexibilidad, sino también un control preciso en el análisis de paquetes.

Esta solución se despliega con el propósito específico de realizar análisis exhaustivos en una red. Desde la detección de anomalías en redes e infraestructuras hasta la identificación de vulnerabilidades, esta herramienta abarca un espectro amplio de funcionalidades. Además, su capacidad para obtener estadísticas detalladas del tráfico consolida su posición como una opción versátil y potente en la gestión y control de redes informáticas.

Definiciones:

Wireshark
Herramienta de análisis de tráfico de red de código abierto que posibilita la captura, inspección y análisis detallado de paquetes de datos, permitiendo una comprensión profunda del flujo de información en una red.[2]

Streamlit
Marco de desarrollo de código abierto que simplifica la creación de aplicaciones web interactivas, utilizado específicamente para la visualización clara y accesible de datos provenientes de análisis de tráfico de red.[4]

Pyshark
Biblioteca de Python especializada en la interacción con datos capturados por Wireshark, ofreciendo flexibilidad y control para el análisis avanzado de paquetes de red[3]

Pandas
Biblioteca de Python centrada en la manipulación y análisis de datos, utilizada en la preparación eficiente de datos capturados por Wireshark para su posterior análisis y visualización.[5]

2 Instrumentación

Para el desarrollo de este trabajo utilizaremos diferentes librerías de Python como también Wireshark.

Configuración de Wireshark
Wireshark, siendo una herramienta fundamental en el análisis de tráfico de red, requiere una configuración adecuada para garantizar una captura efectiva de paquetes. A continuación, se detallan los pasos para configurar Wireshark:

Descarga e Instalación:

- Visitar la página oficial de Wireshark para descargar la versión compatible con el sistema operativo.
- Instalar Wireshark siguiendo las instrucciones proporcionadas durante el proceso de instalación.

Configuración de Interfaces:

Abrir Wireshark y seleccionar la interfaz de red relevante para la captura.
Configurar opciones como la velocidad de captura y los filtros de paquetes según los requisitos específicos.

Inicio de Captura:
Iniciar la captura de paquetes haciendo clic en el botón "Inicio" o utilizando comandos de captura específicos si se está utilizando la interfaz de línea de comandos.
Con esta configuración, Wireshark está listo para capturar y analizar el tráfico de red de manera efectiva.
Ahora, pasemos a la configuración de Pyshark y Pandas para la preparación de datos.

Configuración de Pyshark y Pandas
La combinación de Pyshark y Pandas proporciona una potente herramienta para la preparación y análisis de datos capturados por Wireshark. Aquí se detallan los pasos para su configuración:

Instalación de Pyshark y Pandas:
Utilizar el gestor de paquetes de Python, como pip, para instalar Pyshark y Pandas ejecutando los comandos:

- pip install pyshark
- pip install pandas

Interacción con Datos Capturados:
Importamos las librerías para poder utilizarlas. Pyshark para acceder a los datos capturados por Wireshark y convertirlos en estructuras de datos manipulables por Pandas.
Utilizar Pandas para realizar operaciones de limpieza, filtrado y análisis en los datos, preparándolos para su visualización.
Con esta configuración, Pyshark y Pandas están listos para procesar los datos capturados y extraer información significativa.

Configuración de Streamlit para la Visualización de Datos
Streamlit facilita la creación de aplicaciones web interactivas, brindando una plataforma efectiva para visualizar los datos analizados. A continuación, se describen los pasos para configurar Streamlit:

Instalación de Streamlit:
Utilizar el gestor de paquetes de Python para instalar Streamlit ejecutando el siguiente comando:

- pip install streamlit

Desarrollo de la Aplicación Streamlit:
Crear un script de Python que utilice Streamlit para la visualización de datos.
Importar las bibliotecas necesarias y cargar los datos preparados con Pandas.
Utilizar las funciones proporcionadas por Streamlit para crear gráficos interactivos y presentar los resultados de manera intuitiva.

Inicio de la Aplicación:
Ejecutar el script de Streamlit desde la línea de comandos para iniciar la aplicación web.

- streamlit run app.py

Acceder a la aplicación a través del navegador web en el localhost puerto 8501 para interactuar con los resultados visualizados.

Con esta configuración, Streamlit está preparado para convertir los datos analizados en visualizaciones interactivas y accesibles.

3 Desarrollo

Script de captura de datos y creación de DataFrame

En la fase inicial del desarrollo del sistema, se ha programado un script en Python con

el objetivo de gestionar, capturar y generar un DataFrame a partir de la captura de paquetes de tráfico en red utilizando Wireshark o Pyshark. Este script se ha diseñado para ser una herramienta versátil que proporciona una base sólida para el análisis posterior.

El script define una clase denominada AnalizadorDeRed, la cual facilita la captura de paquetes de tráfico en una interfaz de red específica y con un filtro de captura opcional. Utilizando Pyshark, la clase recopila la información relevante de cada paquete capturado, incluyendo el número de paquete, la marca de tiempo, las direcciones de origen y destino, el protocolo, la longitud y detalles adicionales.

Como parte integral del proceso, se ha incorporado la capacidad de crear un DataFrame de Pandas a partir de los datos capturados.[6] Este DataFrame estructura la información de los paquetes en columnas, facilitando su manipulación y análisis posterior.

Un ejemplo de uso del script se presenta al final del mismo, donde se instancia la clase AnalizadorDeRed, se realiza la captura de 300 paquetes en la interfaz 'WiFi' y se crea un DataFrame que posteriormente se guarda como un archivo CSV denominado 'trafico.csv'. Este archivo se convertirá en la fuente de datos para la visualización en Streamlit.

Este primer paso sienta las bases para el análisis detallado del tráfico de red, permitiendo la preparación de datos necesaria para la visualización en la siguiente etapa del desarrollo.

Vista con Streamlit

En la siguiente fase del desarrollo, se ha creado una interfaz de usuario utilizando Streamlit para visualizar de manera clara los datos generados por el analizador de tráfico en la etapa previa. Streamlit ofrece una solución eficaz y sencilla para implementar interfaces gráficas interactivas, permitiendo la exploración detallada de la información contenida en el DataFrame generado.

La interfaz proporciona diversas visualizaciones del DataFrame, permitiendo obtener información más granular sobre el tráfico de red. Algunas de las funcionalidades clave de la vista en Streamlit incluyen:

Interfaces de Red:
Presentación de interfaces de red disponibles en el host junto con su mac address y dirección ipv4 e ipv6.

IP Destino y Origen:
Visualización de la distribución de paquetes según las direcciones IP de origen y destino, identificando patrones de comunicación.

Protocolos de Capa de Transporte y Aplicación:
Análisis detallado de los protocolos utilizados en las capas de transporte y aplicación, brindando información sobre servicios más ulizados.

Intercambios de Paquetes por IP:
Seguimiento de los intercambios de paquetes asociados a una dirección IP específica, facilitando la comprensión de las interacciones entre nodos de la red.

Consumo de la Red a lo Largo de la Captura:
Representación gráfica del consumo de la red a lo largo del tiempo, permitiendo identificar picos de actividad y tendencias.

La implementación de estas visualizaciones en Streamlit proporciona a los usuarios una plataforma interactiva e intuitiva para explorar y comprender de manera efectiva los datos recopilados durante la captura de tráfico de red. Esta herramienta no solo simplifica el proceso de análisis, sino que también facilita la identificación de patrones y tendencias significativas en el comportamiento de la red.

Este paso en el desarrollo del sistema fortalece la capacidad de obtener percepciones valiosas a partir de los datos capturados, allanando el camino para la toma de decisiones informadas en la gestión y optimización de la red.

4 Ejecución del Programa

Captura de Paquetes con Wireshark y Análisis con Pyshark.

Para ejecutar el programa y generar el archivo .csv que vamos a utilizar para el análisis lo primero que debemos hacer el correr el script analizador.py y esto nos va a abrir una ventana de diálogo donde vamos a poder elegir realizar una captura del tráfico en tiempo real o cargar un archivo .pcap de una captura de paquetes con Wireshark.

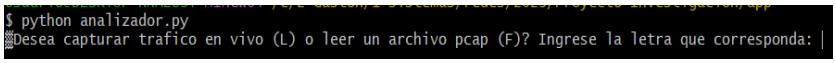

Figura.1

Sin importar la opción que se elija el analizador utiliza Pyshark para cargar el archivo pcap o generar la captura en vivo y extraer los atributos relevantes de cada paquete capturado.

Se crea un DataFrame con Pandas utilizando los atributos extraídos con Pyshark y se los guarda como un archivo CSV con el nombre "trafico.csv".

```python
def capturar_trafico(self, interfaz='Wi-Fi', filtro=None, cantidad_paquetes=300, path_df="tmp\df_trafico_stream.csv"):
    captured_data = []

    cap = pyshark.LiveCapture(interface=interfaz, display_filter=filtro)

    for packet in cap.sniff_continuously(packet_count=cantidad_paquetes):
        timestamp = float(packet.sniff_timestamp)
        formatted_time = datetime.utcfromtimestamp(timestamp).strftime('%Y-%m-%d %H:%M:%S')

        # Obtener el hostname del paquete DNS
        hostname_dns = self.obtener_hostname_dns(packet)
        # Obtener credenciales HTTP
        http_dict = self.obtener_credenciales_http(packet)

        captured_data.append({
            'Timestamp': packet.sniff_timestamp,
            'Date': formatted_time,
            'Source IP': packet.ip.src,
            'Destination IP': packet.ip.dst,
            'Transport Protocol': packet.transport_layer,
            'Application Protocol': packet.highest_layer,
            'Hostname DNS': hostname_dns,
            'HTTP_Protocol_text_plain': http_dict['HTTP_Protocol_text_plain'],
            'HTTP_Protocol_Username': http_dict['HTTP_Protocol_Username'],
            'HTTP_Protocol_Password': http_dict['HTTP_Protocol_Password']
        })

    df = pd.DataFrame(captured_data)
    df.to_csv(path_df, index=False)
    return captured_data
```

Figura.2

```python
def cargar_archivo_pcap(self, pcap_path:str="",path_df="tmp\df_trafico.csv"):
    captured_data = []

    # Crear la instancia de FileCapture
    cap = pyshark.FileCapture(input_file=pcap_path, tshark_path="tshark")


    for packet in cap:
        timestamp = float(packet.sniff_timestamp)
        formatted_time = datetime.utcfromtimestamp(timestamp).strftime('%Y-%m-%d %H:%M:%S')

        # Obtener el hostname del paquete DNS
        hostname_dns = self.obtener_hostname_dns_pcap(packet)

        # Obtener credenciales HTTP
        http_dict = self.obtener_credenciales_http(packet)

        # Obtener información de capa IP
        source_ip, destination_ip = self.obtener_info_ip(packet)

        captured_data.append({
            'Timestamp': packet.sniff_timestamp,
            'Date': formatted_time,
            'Source IP': source_ip,
            'Destination IP': destination_ip,
            'Transport Protocol': packet.transport_layer,
            'Application Protocol': packet.highest_layer,
            'Hostname DNS': hostname_dns,
            'HTTP_Protocol_text_plain': http_dict['HTTP_Protocol_text_plain'],
            'HTTP_Protocol_Username': http_dict['HTTP_Protocol_Username'],
            'HTTP_Protocol_Password': http_dict['HTTP_Protocol_Password']
        })


    df = pd.DataFrame(captured_data)
    df.to_csv(path_df, index=False)
    return captured_data
```

Figura.3

Este proceso garantiza que los datos capturados se analicen de manera efectiva y se almacenen en un formato estructurado para su posterior visualización con Streamlit.

Interfaz de Streamlit

La ejecución del programa continúa con la interfaz de Streamlit, donde se realiza la visualización interactiva de los datos capturados. A continuación, se describen los pasos para esta fase:

- en consola ejecutamos *streamlit run app.py*

Carga del Archivo CSV:

En la interfaz de Streamlit, utilizar la opción de carga para subir el archivo CSV generado durante la captura de paquetes. Este archivo contiene los datos estructurados necesarios para el análisis.

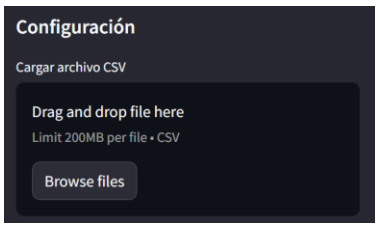

Figura.4

Menú de Opciones en el Sidebar:

Una vez cargado el archivo CSV, se despliega un menú en el sidebar con diversas opciones de filtro para explorar los datos de manera específica.

- Interfaz de Red: Filtrar por una interfaz de red específica utilizada en la captura.
- IP de Origen y Destino: Filtrar por direcciones IP de origen y destino.
- Protocolo de Capa de Transporte: Seleccionar un protocolo de la capa de transporte específico.

- Protocolo de Capa de Aplicación: Filtrar por protocolo de la capa de aplicación.
- Número de Paquetes a Visualizar: Definir la cantidad de paquetes que se mostrarán en las visualizaciones.
- Fecha y Hora: Filtrar por un rango específico de fecha y hora.

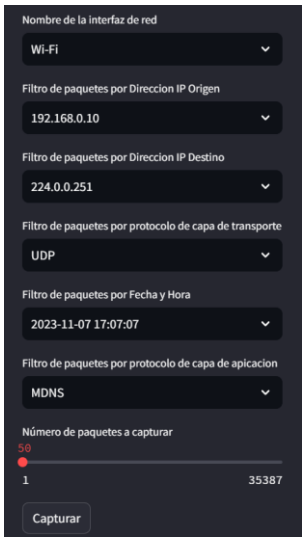

Figura.5

Estos filtros permiten una exploración detallada y personalizada de los datos capturados, proporcionando flexibilidad en el análisis de distintos aspectos del tráfico de red.

Este proceso en Streamlit, desde la carga del archivo CSV hasta la aplicación de filtros específicos, facilita una visualización interactiva y detallada de los resultados obtenidos del análisis de tráfico de red.

Visualización en Streamlit

La visualización en Streamlit se destaca por su capacidad de presentar datos de manera clara y accesible. Después de aplicar los filtros específicos en el sidebar, la interfaz ofrece dos pestañas distintas para explorar los datos capturados:

Pestaña "Análisis de Tráfico de Red":

En esta sección, se presenta un análisis detallado con diferentes DataFrames, cada uno correspondiente al tipo de filtro aplicado. Estos DataFrames proporcionan información específica según el filtro seleccionado, permitiendo una exploración más profunda de los datos. Algunos ejemplos incluyen:

- DataFrames filtrados por interfaz de red.
- DataFrames para direcciones IP de origen y destino específicas.
- DataFrames según el protocolo de capa de transporte y aplicación

Pestaña "Gráficos":

Esta pestaña ofrece visualizaciones gráficas que facilitan la comprensión rápida e intuitiva del tráfico de paquetes:

- Gráficos de Flujo de Paquetes desde una IP: Muestra la distribución del tráfico desde una dirección IP específica hacia el resto de la red.
- Obtención de Nombres de Dominio desde una IP: Permite identificar el nombre de dominio asociado a una dirección IP dada.
- Análisis del tráfico de la red en el tiempo: Representación gráfica del consumo de la red a lo largo del tiempo, permitiendo identificar picos de actividad y tendencia.

Estas pestañas en Streamlit ofrecen una experiencia integral de análisis, permitiendo a los usuarios explorar datos tabulares detallados y comprender visualmente el comportamiento del tráfico de red a través de gráficos intuitivos.

5 Resultados

En esta sección, se presentan algunos resultados clave obtenidos del análisis del tráfico de red utilizando la herramienta desarrollada. Los datos filtrados y los patrones identificados proporcionan una visión detallada de la actividad en la red durante la captura.

Interfaces.

La pestaña interfaces permite visualizar las interfaces de red disponibles en el host junto con su dirección mac address, ipv4 e ipv6 y la máscara.

Figura.6

Capturas de Datos Filtrados:

A continuación se muestran fragmentos de DataFrames correspondientes a diferentes filtros aplicados durante el análisis de tráfico de red. Estos ejemplos ilustran la capacidad de la herramienta para proporcionar información específica según criterios como interfaz de red, direcciones IP, y protocolos de capa de transporte y aplicación.

	Timestamp	Date	Source IP	Destination IP	Transport Protocol
0	1,699,376,827.5858	2023-11-07 17:07:07	192.168.0.10	224.0.0.251	UDP
1	1,699,376,827.5864	2023-11-07 17:07:07	fe80::f78d:6837:a14a:d81e	ff02::fb	UDP
2	1,699,376,829.8859	2023-11-07 17:07:09	None	None	None
3	1,699,376,829.8878	2023-11-07 17:07:09	None	None	None
4	1,699,376,829.9502	2023-11-07 17:07:09	192.168.0.10	181.30.140.198	UDP
5	1,699,376,829.9735	2023-11-07 17:07:09	181.30.140.198	192.168.0.10	UDP
6	1,699,376,830.9741	2023-11-07 17:07:10	192.168.0.10	181.30.140.198	UDP
7	1,699,376,830.9879	2023-11-07 17:07:10	192.168.0.10	181.30.140.198	UDP
8	1,699,376,830.9974	2023-11-07 17:07:10	181.30.140.198	192.168.0.10	UDP
9	1,699,376,831.0126	2023-11-07 17:07:11	181.30.140.198	192.168.0.10	UDP

Figura.7

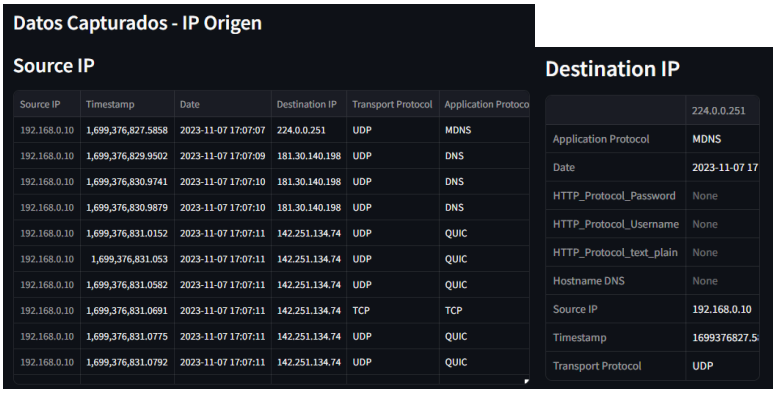

Figura.8

Figura.9

Análisis Temporales:

Los gráficos temporales revelan patrones de actividad a lo largo del tiempo de la captura de tráfico. Estas representaciones ofrecen insights sobre picos de actividad, períodos de inactividad y tendencias generales en la transmisión de paquetes.

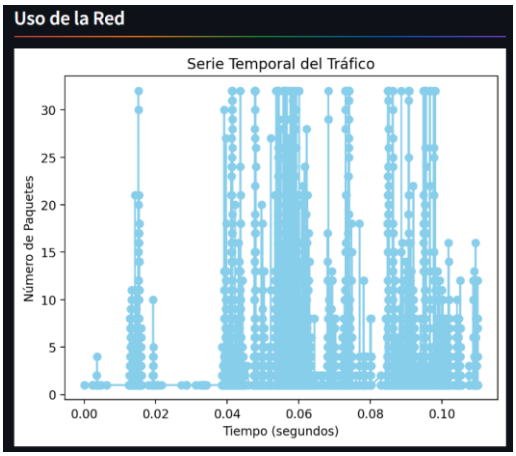

Figura.10

Patrones Identificados:

Al analizar el flujo de paquetes desde una IP específica, se revelan patrones de comunicación. Los gráficos destacan las direcciones IP más frecuentemente contactadas desde la IP de origen seleccionada, brindando una comprensión clara de las interacciones dominantes.

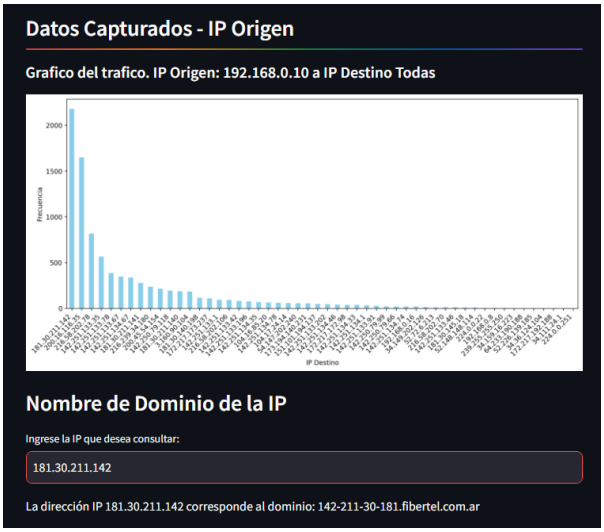

Figura.11

Estos resultados proporcionan una visión integral de la actividad de la red, permitiendo identificar patrones específicos, tendencias temporales y relaciones entre nodos de la red. La herramienta desarrollada se demuestra eficaz en el análisis detallado y la presentación visual de datos capturados, facilitando la toma de decisiones informadas en la gestión y optimización de la red.

Puede consultar la implementación y probarla en: analizador_pyshark [7]

6 Conclusión y posibles mejoras

La experiencia académica de desarrollar y aplicar esta herramienta para el análisis de tráfico de red ha sido enriquecedora y ha arrojado resultados significativos. Se logró cumplir con el objetivo principal de crear una herramienta que permite un análisis rápido, simple e intuitivo del tráfico de red. La utilización de librerías de código abierto ha demostrado ser una elección acertada, brindando flexibilidad y economía, en comparación con soluciones de mercado pagas.

La implementación de Pyshark y Pandas ha permitido una manipulación eficiente de los datos capturados, facilitando la creación de visualizaciones detalladas en Streamlit. La capacidad de aplicar diversos filtros ha proporcionado a los usuarios la flexibilidad necesaria para explorar aspectos específicos del tráfico de red, desde interfaces y direcciones IP hasta protocolos de transporte y aplicación.

Posibles Mejoras

Análisis de Paquetes HTTP:

Integrar el análisis de paquetes HTTP para identificar y filtrar datos sensibles, mejorando la capacidad de la herramienta para detectar potenciales vulnerabilidades y amenazas.

Análisis de Tráfico a Puertos:

Incluir la capacidad de analizar el tráfico de paquetes enviados a puertos específicos, permitiendo la identificación de posibles ataques dirigidos a servicios específicos.

Visualización en Tiempo Real:

Implementar la visualización en tiempo real, permitiendo que los gráficos se actualicen dinámicamente durante la ejecución de los paquetes. Esta mejora proporciona una perspectiva en tiempo real del estado actual de la red.

7 Referencias

[1] Cisco Secure Network Analytics vs SolarWinds NPM. (n.d.). TrustRadius. Retrieved November 8, 2023, from https://www.trustradius.com/compare-products/cisco-secure-network-analytics-vs-solarwinds-network-performance-monitor#pricing

[2] Wireshark. (n.d.). Wireshark. Retrieved November 8, 2023, from https://www.wireshark.org/

[3] KimiNewt/pyshark: Python wrapper for tshark, allowing python packet parsing using wireshark dissectors. (n.d.). GitHub. Retrieved November 8, 2023, from https://github.com/KimiNewt/pyshark/

[4] Streamlit. (n.d.). Streamlit • A faster way to build and share data apps. Retrieved November 8, 2023, from https://streamlit.io/

[5] Pandas - Python Data Analysis Library. (n.d.). Retrieved November 8, 2023, from https://pandas.pydata.org/

[6] Pandas. (n.d.). pandas.DataFrame — pandas 2.1.2 documentation. Retrieved November 10, 2023, from https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

[7] Escalante, G. (2023, Noviembre 10). Analizador PyShark. https://github.com/gaston411/analizador_pyshark. Retrieved November 10, 2023, from https://github.com/gaston411/analizador_pyshark