Created by Yuvin Perera

Wireshark Script Documentation

Purpose of the Scripts

Wireshark is a popular network protocol analyzer used for troubleshooting, analysis, and security audits. TShark, its command-line counterpart, provides powerful scripting capabilities. The provided scripts automate tasks such as capturing packets, analyzing packet statistics, and extracting HTTP requests, making it easier to perform network analysis efficiently.

The first script, Packet Capture with Filters, captures packets on a specified network interface and applies a filter to focus on specific traffic types (e.g., TCP, HTTP). The second script, Packet Statistics, generates an overview of traffic patterns from a capture file, including protocols and their counts. The third script, Extract HTTP Requests, extracts HTTP-related information from a capture file, such as hosts, URIs, and user agents, and saves it to a text file for analysis.

Usage Instructions

General Setup

1. Ensure Wireshark and TShark are installed. You can download them from the Wireshark website.
2. Verify TShark is accessible from your terminal or command prompt by running:
3. tshark --version
4. Install Python if not already installed.

Script 1: Packet Capture with Filters

Command:

```
python packet_capture.py
```

Required Parameters:

- ✓ interface: The network interface to capture packets from (e.g., eth0, wlan0).
- ✓ duration: The duration of the capture in seconds (e.g., 60).
- ✓ filter_expr: The packet filter expression (e.g., tcp port 80).

Script 2: Packet Statistics

Command:

```
python packet_statistics.py
```

Required Parameters:

- ✓ input_file: The path to the packet capture file (e.g., capture.pcap).

Script 3: Extract HTTP Requests

Command:

```
python extract_http_requests.py
```

Required Parameters:

- ✓ input_file: The path to the packet capture file (e.g., capture.pcap).
- ✓ output_file: The path to save the extracted HTTP request details (e.g., http_requests.txt).

Expected Outputs or Results

Packet Capture with Filters

- ✓ Captures packets matching the specified filter and saves them to a .pcap file.
- ✓ Outputs a confirmation message with the file name.

Packet Statistics

- ✓ Analyzes a .pcap file and prints traffic statistics, such as the number of packets for each protocol.

Extract HTTP Requests

- ✓ Extracts HTTP request details (hosts, URIs, user agents) from a .pcap file and saves them in a text file.

Dependencies Needed

- ✓ Wireshark/TShark: Required for packet capturing and analysis.
- ✓ Python: Required to run the scripts.

Line-by-Line Explanation

Packet Capture with Filters

import subprocess

- subprocess: Used to run TShark commands from the script.

def capture_packets(interface, duration, filter_expr):

- Defines a function to capture packets with specified parameters.

output_file = f"capture_{interface}_{duration}s.pcap"

- Constructs the output file name based on the interface and duration.

subprocess.run(["tshark", "-i", interface, "-a", f"duration:{duration}", "-f", filter_expr, "-w", output_file])

- Executes the TShark command to capture packets with the specified interface, duration, and filter expression.

print(f"Capture complete. Results saved to {output_file}")

- Prints a message confirming the capture is complete.

Packet Statistics

import subprocess

- subprocess: Used to run TShark commands for analyzing packet statistics.

def packet_statistics(input_file):

- Defines a function to analyze packet statistics from a .pcap file.

subprocess.run(["tshark", "-r", input_file, "-q", "-z", "io,phs"])

- Runs TShark with the -z io,phs option to display packet hierarchy statistics.

print("Analysis complete.")

- Prints a message indicating the analysis is complete.

Extract HTTP Requests

import subprocess

- subprocess: Used to run TShark commands for extracting HTTP requests.

def extract_http_requests(input_file, output_file):

- Defines a function to extract HTTP requests and save them to a file.

subprocess.run(["tshark", "-r", input_file, "-Y", "http.request", "-T", "fields", "-e", "http.host", "-e", "http.request.uri", "-e", "http.user_agent", "-w", output_file])

- Runs TShark with filters and field options to extract HTTP-related data.

print(f"HTTP requests extracted and saved to {output_file}")

- Prints a message confirming the extraction is complete.

Why These Scripts Are Needed ?

Automating Wireshark tasks reduces the effort and complexity involved in network analysis. The Packet Capture with Filters script allows targeted traffic capture, minimizing unnecessary data and focusing on specific protocols or ports. The Packet Statistics script provides an overview of the traffic composition, helping analysts identify patterns and anomalies. The Extract HTTP Requests script simplifies the extraction of valuable HTTP data, enabling detailed analysis of web traffic for troubleshooting or security assessments.