

K-9 Packet Sniffer – Backend Development

Students: Carlos Imery

Instructor/Faculty: *Dr. Seyedmasoud Sadjadi*, Florida International University

PROBLEM

Network administrators and security professionals often rely on tools like Wireshark to capture and analyze network traffic.

However, the absence of user-based session management in these tools is a critical issue, as it leads to challenges such as:

Manual File Management: Users must manually save, organize, and track capture files.

No built-in user authentication: Securely associating sessions with specific individuals is challenging, highlighting the need for robust user access control.

Risk of Data Loss or Mismanagement: Without proper organization, session data is at risk of being misplaced or mishandled, posing as a security concern.

CURRENT SYSTEM

Existing tools:

- Wireshark – Network Analysis and packet capture in real-time.

Manual Session Management: Users can only download in save capture files themselves, leading to potential disorganization and data loss.

No User-Based Access Control: Anyone with access to the system can view or manipulate saved session files.

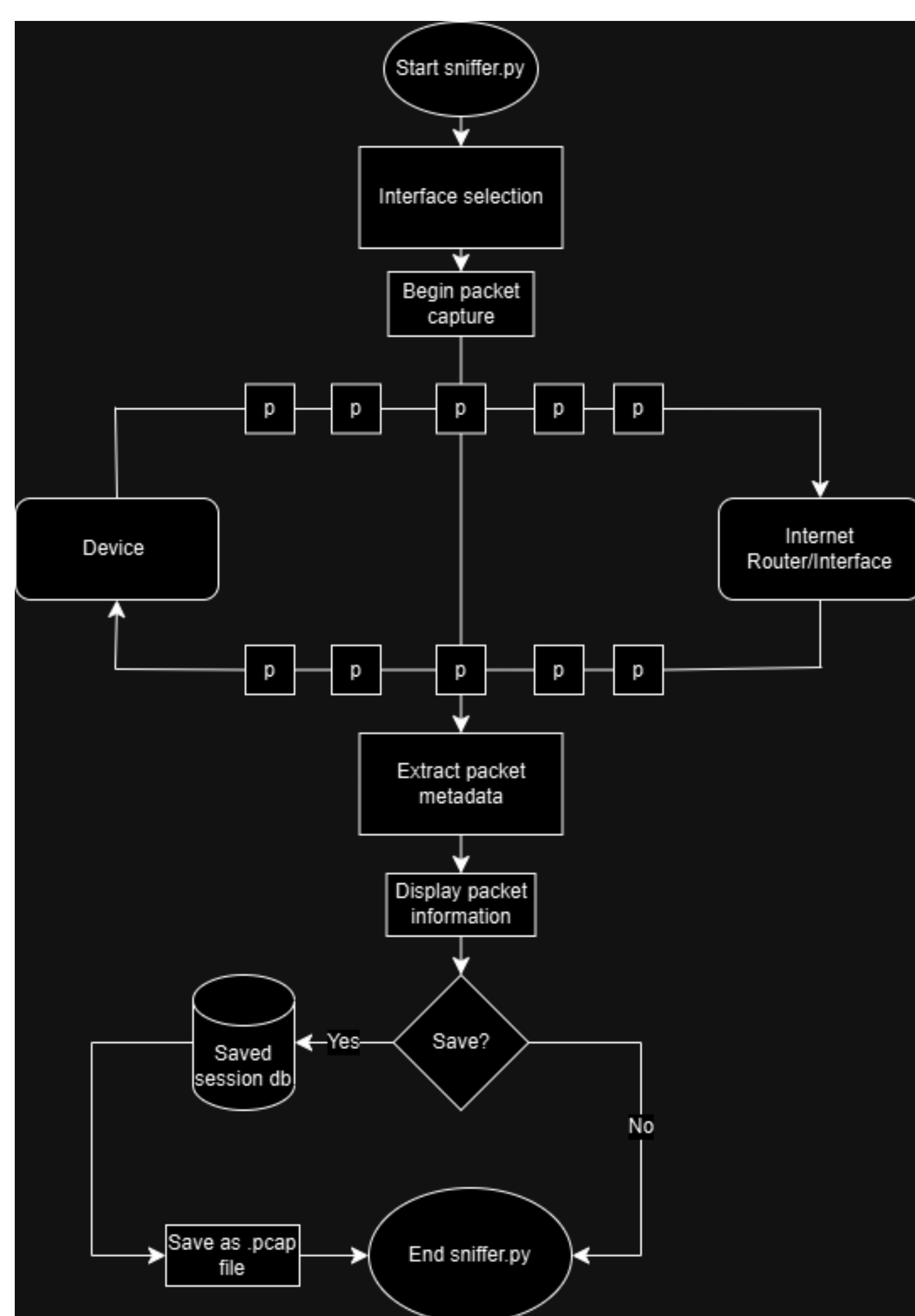
Scalability Issues: Managing numerous capture files becomes cumbersome, especially in multi-server environments.

Steep Learning Curve: The interface can be overwhelming for beginners, and certain features lack user-friendly implementation.

REQUIREMENTS



SNIFFER PROTOCOL



UTILIZATION / REQUIREMENTS

- Note: This is the code prior to being adapted to function alongside the front end. Functionality is the same, but output structure differs.**

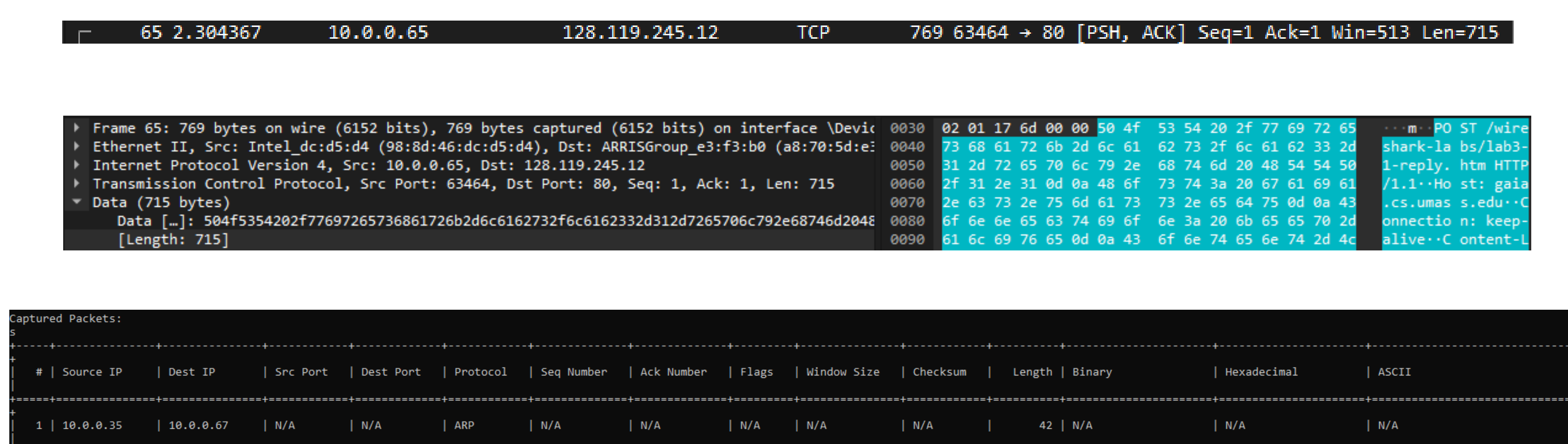
- Requirements (basic):
 - Python 3
 - Scapy 2.6.1 - Scapy is a powerful interactive packet manipulation library written in Python.
 - Psutil 6.1.0 - A cross-platform library for retrieving information on running processes and system utilization (CPU, memory, disks, network, sensors) in Python.

- Code run on terminal:

#	Source IP	Dest IP	Src Port	Dest Port	Protocol	Seq Number	Ack Number	Flags	Window Size	Checksum	Length	Binary	Hexadecimal	ASCII
1	10.0.0.35	10.0.0.67	N/A	N/A	ARP	N/A	N/A	N/A	N/A	N/A	42	N/A	N/A	N/A

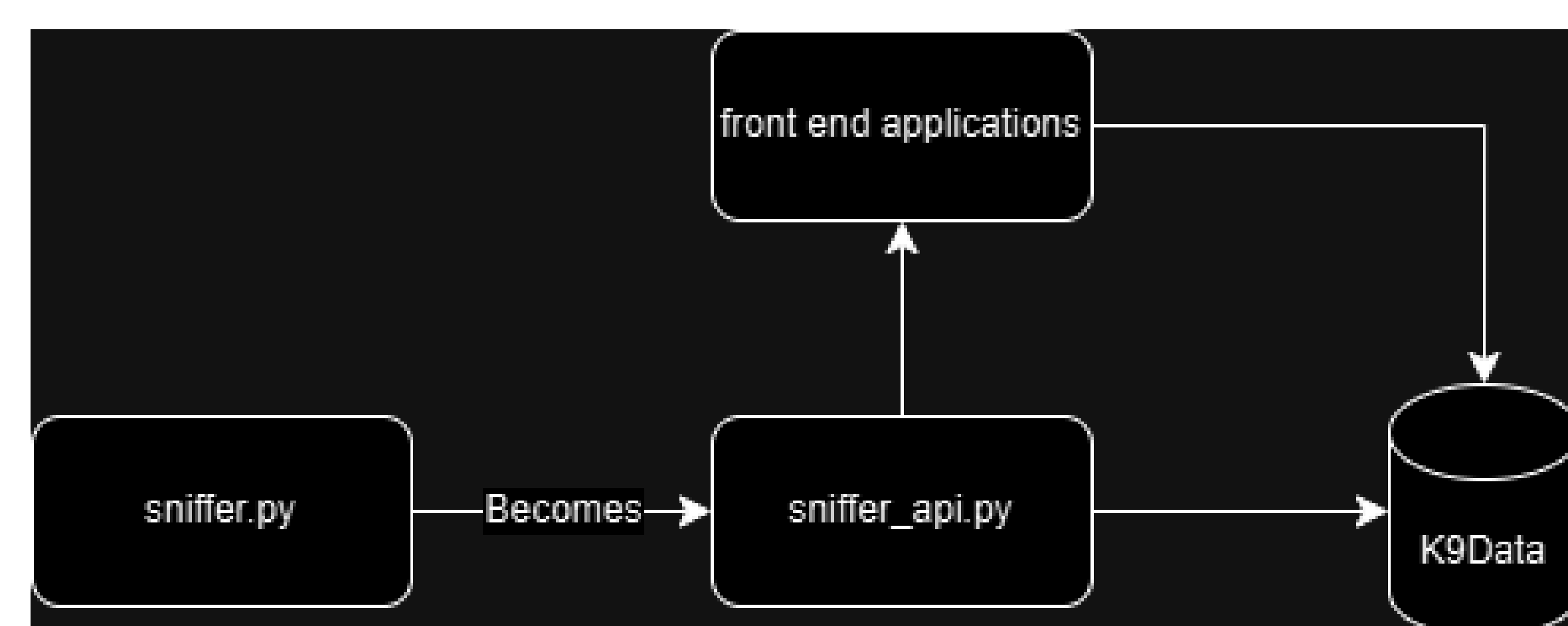
VERIFICATION

In order to verify the validity, and accuracy of our packet sniffer, we ran the sniffer alongside a Wireshark process, to verify that the captured packets match each other. We do this as we still recognize Wireshark as a reliable tool for packet captures, it just lacks some security and organizational features.



SUMMARY

- Sniffer.py takes care of the processes that intercept and read package data; it becomes sniffer_api.py when molded to work with the front-end applications, and it is given access to the K9Data database as we store the individual packets captured and capture sessions for log keeping purposes.



Note: Simplified Diagram