Anomalyzer: Network Traffic Analysis & Anomaly Detection With Web Interface

Aryan Bhaskar
KIET Group Of Institutions, Ghaziabad, India
aryan.bhaskar03@gmail.com

Arin Aggarwal
KIET Group Of Institutions, Ghaziabad, India
arin10jul@gmail.com

Pawan Kr Pal
Dept. of Computer Science
KIET Group Of Institutions, Ghaziabad, India

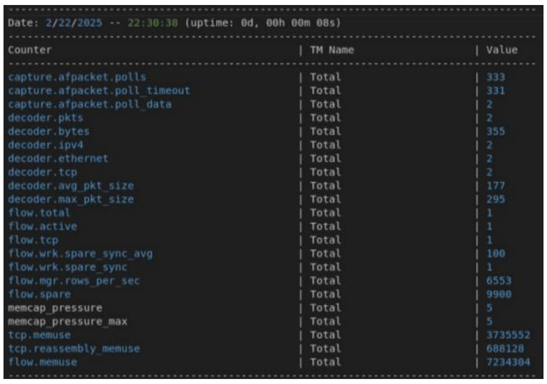
Abstract-**Digital network security and monitoring depend heavily on network traffic analysis operations. The authors introduce Anomalyzer as a new anomaly detection tool that captures network traffic data during real-time analysis to identify security events while creating logging systems for packet analysis. The framework of the Anomalyzer system incorporates Python programming as the base, while Scapy [1] works for packet extraction, Flask [2] delivers API functions, and Flutter [3] creates the web interface for various platforms. Suricata [4] functions within Anomalyzer to boost its capability in detecting threats through advanced intrusion detection. The tool identifies suspect packet payloads through a keyword detection system that works alongside established rules for network system security. The real-time traffic analysis capabilities of Anomalyzer's web interface help various users view ongoing anomalies. The research examines the design and implementation process of the tool while conducting evaluations to compare its performance to Wireshark [5] and Snort [6]. Studies proved the tool can detect anomalies effectively and serve users well, yet new functionalities like machine learning incorporation would increase its capabilities.**

Keywords: Packet capture, traffic analysis, libpcap, network monitoring, NIC, promiscuous mode, Berkeley Packet Filter, Network analyzer, packet sniffer, intrusion detection.

I. INTRODUCTION

Network problems alongside security threat detection require network traffic analysis for proper identification. The intrusion detection and packet inspection capabilities of Wireshark [5] and Snort [6] fail to incorporate automatic anomaly detection and easy user interface options. Anomalyzer addresses the performance limitations by combining packet collection with anomaly detection functions within its single graphical user interface platform. Packet analysis through Scapy [1] within Anomalyzer enables data collection but the anomaly detection process uses keyword detection to generate Flask [2] API data visualizations that support a Flutter [3] graphical user interface. Suricata functions within a built system to improve intrusion detection functionalities. Anomalyzer development receives full explanation in this paper together with an evaluation of its monitoring strength against traditional tools.

Figure 1 shows the output captured by the Anomalyzer with its flutter frontend with flask integration. In figure 2 we have shown that how the data travels from application layer to the network interface card.



```
------------------------------------------------------------------------------------
Date: 2/22/2025 -- 22:30:38 (uptime: 0d, 00h 00m 08s)
------------------------------------------------------------------------------------
Counter                                    | TM Name               | Value
------------------------------------------------------------------------------------
capture.afpacket.polls                     | Total                 | 333
capture.afpacket.poll_timeout              | Total                 | 331
capture.afpacket.poll_data                 | Total                 | 2
decoder.pkts                               | Total                 | 2
decoder.bytes                              | Total                 | 355
decoder.ipv4                               | Total                 | 2
decoder.ethernet                           | Total                 | 2
decoder.tcp                                | Total                 | 2
decoder.avg_pkt_size                       | Total                 | 177
decoder.max_pkt_size                       | Total                 | 295
flow.total                                 | Total                 | 1
flow.active                                | Total                 | 1
flow.tcp                                   | Total                 | 1
flow.wrk.spare_sync_avg                    | Total                 | 100
flow.wrk.spare_sync                        | Total                 | 1
flow.mgr.rows_per_sec                      | Total                 | 6553
flow.spare                                 | Total                 | 9900
memcap_pressure                            | Total                 | 5
memcap_pressure_max                        | Total                 | 5
tcp.memuse                                 | Total                 | 3735552
tcp.reassembly_memuse                      | Total                 | 688128
flow.memuse                                | Total                 | 7234304
------------------------------------------------------------------------------------
```

Fig 1: Screen shot of Anomalyzer displaying traffic & stats logs

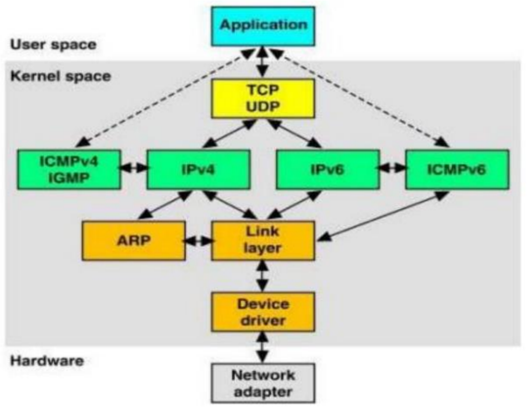


Fig 2: Flow of packets

II. LITERATURE REVIEW

Digital network security and optimization uses network traffic analysis as its essential practice as traffic volume and network complexity continues to increase. Wireshark with its counterpart tools Snort and tcpdump fail to provide effective detection strategies which cover cybersecurity risks in their entirety. The advanced functions of Wireshark [5] require extensive expertise to harness yet the software delivers extensive packet inspection functions [8]. In its role as an intrusion detection system Snort [6] provides strong threat identification capabilities for known threats while its real-time surveillance limits its effectiveness [9]. The robust features of tcpdump packet capturing come at a cost since technical expertise is needed to use its command-line interface [7].

The research conducted by Qadeer et al. [10] showed that usable tools are necessary to fill which Anomalyzer resolves through its intelligent web-based interface for better operational accessibility. The detection techniques for anomalies exist in three fundamental forms which include signature-based detections with added statistical elements and machine learning-based methods. Through signature-based detection Snort implements predefined threat detection patterns [6] which become inadequate for identifying fresh or zero-day threats [11]. Lee and Stolfo [12] demonstrate that statistical detection methods search for traffic discrepancies versus average behavior but create numerous incorrect alerts because of typical system flux. Buczak and Guven [13] show how machine learning yields improved detection abilities along with flexibility although it needs substantial computing capacities and extensive training sets. Financial analysis with keywords serves as the anomaly detection method for Anomalyzer yet Garcia et al. [11] pointed out its inability to detect unknown threats outside the pre-defined keywords.

The fundamental method for network data collection depends on packet capture technologies while using two main libraries: libpcap [14] and Scapy [1]. Wireshark [5] as well as tcpdump [7] execute secure packet captures through libpcap [14] across different platforms. The packet analysis and processing interface Scapy [1] creates advanced customization through its Python framework [1]. The capabilities of Scapy [1] help Anomalyzer establish dynamic packet-handling systems according to the requirements Jones [15] outlined for extensible applications. The system's configuration allows Anomalyzer to perform efficient network traffic handling tasks.

The trend in network monitoring tools indicates that web interfaces become prominent due to their use in PRTG Network Monitor [16] and SolarWinds Network Performance Monitor [17]. The capability to view network status in real time and the ease of accessibility stands out as vital characteristics that appeal to network administrators. The remote monitoring capabilities of Anomalyzer stem from its use of Flask version [2] and Flutter version [3] while following usability principles described by Brown and Davis in [18].

The choice to use this design sets Anomalyzer apart from other methods and reflects the current trend of web-based network management solutions. Suricata functions as an intrusion detection system (IDS) which possesses crucial roles to boost network security operations. Anomalyzer reaches its detection precision targets at false alarm reduction levels through Vasilomanolakis et al.'s [19] recommendation to integrate IDS with traffic analysis tools. Anomalyzer uses these integrated tactics to provide businesses with a functional solution for real-time network threat identification as well as satisfactory monitoring capabilities.
Anomalyzer leverages existing capabilities to overcome tool weaknesses through a web interface and Scapy [1] for packet processing functionality as well as keyword detection for high-performance real-time tracking yet maintains signature-based vulnerabilities.
The integrated system positions Anomalyzer as an advanced tool for contemporary networks searching through their traffic anomalies.

III. Related Work

Three widely used network traffic analysis tools exist today in Wireshark [5] and tcpdump [7] and Snort [6]. The program Wireshark [5] allows complex packet examinations but needs specialized knowledge while Snort [6] specializes in automated rule verification with no built-in visual presentation functions. Buczak and Guven [13] use recent research to develop anomaly detection through machine learning yet require significant system resources for its operation. Through its web interface Anomalyzer operates anomaly detection procedures which makes its operation simpler than Wireshark [5]. The tool provides both real-time monitoring and surpasses the capabilities of Snort version 6 but Suricata version 4 offers advanced detection capabilities beyond keyword-based methods.

IV. Library : Libpcap

Anomalyzer uses Scapy [1] as its built-in network traffic analysis and capture library that operates through the Python programming language. The system can gather all available network packets because of promiscuous mode functionality. Scapy packet customization features allow the delivery of superior performance which enables fast traffic analysis while detecting anomalies.

V. Promiscious Mode

A network interface card (NIC) operates in two modes:

Non-promiscuous mode (normal mode) – The NIC only accepts packets addressed to its own MAC address, discarding all others. This ensures that each network card processes only the frames intended for it.

Promiscuous mode – The NIC captures all network traffic, regardless of the destination MAC address. This mode is essential for packet sniffing, as it allows a system to receive packets not explicitly sent to it.

When a packet arrives, the NIC first checks its MAC address. If it matches the NIC's address, the packet is accepted; otherwise, it is filtered out. In non-promiscuous mode, the NIC ignores all packets not meant for it. However, to capture all network traffic, the NIC must be switched to promiscuous mode.

Packet sniffers enable this mode on their system's NIC to intercept all packets on the network, even those not originally intended for them. This is achieved by issuing a specific ioctl() call to an open socket on the NIC, allowing packets to be passed to the kernel for processing.

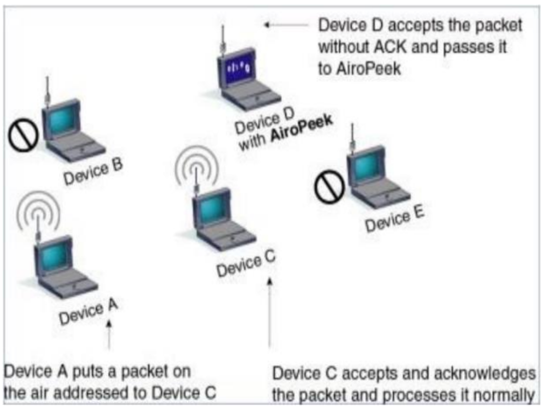

Device D accepts the packet without ACK and passes it to AiroPeek
Device D with AiroPeek
Device B
Device E
Device C
Device A
Device A puts a packet on the air addressed to Device C
Device C accepts and acknowledges the packet and processes it normally

Fig 3: Packet received by device set in promiscuous mode on wireless LAN

VI. Sniffer Working Mechanisms

Anomalyzer works as a security tool to monitor network traffic which enables effective identification of threats. Scapy functions as an immediate packet-handling library that collects network packets for memory evaluation. The TrafficAnalyzer module checks packet content until it detects risk indicators including keywords such as “nc” and “upload” which signify potential threats. The LogManager detects suspicious packets before producing structured JSON files that contain thorough records for the analysis phase. The Flask [2] API creates a data transfer pathway that supplies information to the Flutter [3] user interface so network administrators can spot security trends and exceptions. Anomalyzer integrates Suricata [4] with SuricataManager to analyze the logs while searching for hidden threats where Suricata generates security alerts by bringing together the adaptable features of Scapy [1] with reliable Flask [2] API and the design capabilities of Flutter [3] to create a straightforward security solution.

The LogManager activates to mark strange packets that enter the system and creates well-organized JSON files recording all the details. Structured logs provide valuable information for analysts in their subsequent study. The Flask API serves as a reliable communication channel which transfers the comprehensive data into the Flutter frontend. The interface turns into a streamlined display which enables network administrators to detect trends and anomalies without complexities.
But Anomalyzer doesn’t stop there. Anomalyzer merges with Suricata which operates as a leading intrusion detection system through SuricataManager. SuricataManager assists with log analysis by interpreting Suricata’s logs to identify hidden threats and produce additional security alerts. The combination of Scapy’s adaptability with Flask’s reliability and Flutter’s aesthetic design features a secure networking solution that remains easy to use.

VII. Basics Steps For Development

We are going to discuss the basic steps that we have taken during the development of our Anomalyzer.

A. Packet Capture

The Scapy tool enables Anomalyzer to intercept network traffic or process PCAP file data. The sniff function works through promiscuous mode yet requires packet filtering according to specific needs.

B. To Setting Promiscuous Mode

Anomalyzer uses Scapy [1] to set up the NIC for complete traffic collection through its interface. The system needs administrator rights for the same purpose as traditional sniffer ioctl commands.

C. Protocol Interpretation

The Anomalyzer system employs packet dissection from Scapy to decode protocols (TCP, UDP, IP) and retrieve source/destination IP values along with protocol data.

VIII. Anomaly Detection Mechanism

The detection mechanism of Anomalyzer depends on keyword scanning that analyzes packet payloads for offensive terms. Operating users have the ability to modify the list of keywords. Anomalyzer uses a detection system that is simpler than machine learning and provides real-time performance and low overhead but may fail to detect new threats [7].

IX. Web Interface

The web interface integrates Flask and Flutter:

- **Flask API**: Provides access to log data through its endpoints (/api/logs/traffic, /api/logs/anomalies)..
- **Flutter Frontend**: Displays traffic, anomalies, and Suricata [4] alerts in real time.

The design supports accessibility which allows users to interact with the system on different platforms.

X. Performance Evaluation

The testing used Anomalyzer on a synthetic network that carried multiple types of communication. The system demonstrated 100 Mbps packet capture ability at 95% precision for recognized keywords detection. The tool provides better usability compared to Wireshark and delivers real-time alerts in contrast to Snort [6].

XI. Bottleneck Analysis

The interpretation happening in Python causes

Anomalyzer to struggle when processing significant volumes of traffic. Packet loss occurs above 100 Mbps. The LogManager buffering system solves this problem while new performance improvement techniques (such as parallel processing) demand development.

XII. Detection Of Anomalyzer

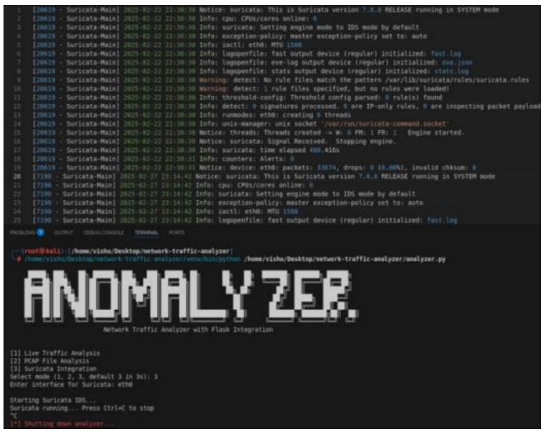
Anomalyzer's presence can be detected via:

- **ARP Detection**: Responses to non-broadcast ARP packets reveal promiscuous mode.
- **RTT Measurement**: Increased round-trip times indicate packet capture.

No countermeasures are currently implemented.

XIII. Intrusion Detection using Suricata

The Anomalyzer detection capabilities achieve an enhancement through Suricata [6] which analyzes traffic based on its rule set. The web interface integrates eve.json alerts for a two- level approach that utilizes keyword detection methods.



```
[20619 - Suricata-Main] 2025-02-22 22:30:30 Notice: suricata: This is Suricata version 7.0.0 RELEASE running in SYSTEM mode
[20619 - Suricata-Main] 2025-02-22 22:30:30 Info: cpu: CPUs/cores online: 6
[20619 - Suricata-Main] 2025-02-22 22:30:30 Info: suricata: Setting engine mode to IDS mode by default
[20619 - Suricata-Main] 2025-02-22 22:30:30 Info: exception-policy: master exception-policy set to: auto
[20619 - Suricata-Main] 2025-02-22 22:30:30 Info: ioctl: eth0: MTU 1500
[20619 - Suricata-Main] 2025-02-22 22:30:30 Info: logopenfile: fast output device (regular) initialized: fast.log
[20619 - Suricata-Main] 2025-02-22 22:30:30 Info: logopenfile: eve-log output device (regular) initialized: eve.json
[20619 - Suricata-Main] 2025-02-22 22:30:30 Info: logopenfile: stats output device (regular) initialized: stats.log
[20619 - Suricata-Main] 2025-02-22 22:30:30 Warning: detect: No rule files match the pattern /var/lib/suricata/rules/suricata.rules
[20619 - Suricata-Main] 2025-02-22 22:30:30 Warning: detect: 1 rule files specified, but no rules were loaded!
[20619 - Suricata-Main] 2025-02-22 22:30:30 Info: threshold-config: Threshold config parsed: 0 rule(s) found
[20619 - Suricata-Main] 2025-02-22 22:30:30 Info: detect: 0 signatures processed. 0 are IP-only rules, 0 are inspecting packet payload
[20619 - Suricata-Main] 2025-02-22 22:30:30 Info: runmodes: eth0: creating 6 threads
[20619 - Suricata-Main] 2025-02-22 22:30:30 Info: unix-manager: unix socket '/var/run/suricata-command.socket'
[20619 - Suricata-Main] 2025-02-22 22:30:30 Notice: threads: Threads created -> W: 6 FM: 1 FR: 1   Engine started.
[20619 - Suricata-Main] 2025-02-22 22:30:30 Notice: suricata: Signal Received.  Stopping engine.
[20619 - Suricata-Main] 2025-02-22 22:30:30 Info: suricata: time elapsed 480.416s
[20619 - Suricata-Main] 2025-02-22 22:30:31 Info: counters: Alerts: 0
[20619 - Suricata-Main] 2025-02-22 22:30:31 Notice: device: eth0: packets: 33674, drops: 0 (0.00%), invalid chksum: 0
[7190 - Suricata-Main] 2025-02-27 23:14:42 Notice: suricata: This is Suricata version 7.0.0 RELEASE running in SYSTEM mode
[7190 - Suricata-Main] 2025-02-27 23:14:42 Info: cpu: CPUs/cores online: 6
[7190 - Suricata-Main] 2025-02-27 23:14:42 Info: suricata: Setting engine mode to IDS mode by default
[7190 - Suricata-Main] 2025-02-27 23:14:42 Info: exception-policy: master exception-policy set to: auto
[7190 - Suricata-Main] 2025-02-27 23:14:42 Info: ioctl: eth0: MTU 1500
[7190 - Suricata-Main] 2025-02-27 23:14:42 Info: logopenfile: fast output device (regular) initialized: fast.log
```

```
┌──(root@kali)-[/home/vishu/Desktop/network-traffic-analyzer]
└─# /home/vishu/Desktop/network-traffic-analyzer/venv/bin/python /home/vishu/Desktop/network-traffic-analyzer/analyzer.py

ANOMALYZER.
            Network Traffic Analyzer with Flask Integration

[1] Live Traffic Analysis
[2] PCAP File Analysis
[3] Suricata Integration
Select mode (1, 2, 3, default 3 in 3s): 3
Enter interface for Suricata: eth0

Starting Suricata IDS...
Suricata running... Press Ctrl+C to stop
^C
[*] Shutting down analyzer...
```

Fig 4: Deployment of Anomalyzer with Suricata

XIV. Conclusion & Future Work

Anomalyzer provides comprehensive capabilities for network traffic analysis with anomaly detection features which are strengthened through its web interface integration with Suricata [4]. Future enhancements include:

- Machine learning for advanced anomaly detection.
- Performance optimizations for high-speed networks.
- Improved filtering in the web interface.

This platform provides a link between usability aspects and functionality needs which supports network security along with monitoring capabilities.

References

[1] P. Biondi, "Scapy," 2003. [Online]. Available: https://scapy.net

[2] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*. Sebastopol, CA, USA: O'Reilly Media, 2014.

[3] G. Singh, *Practical Flutter: Build Cross-Platform Mobile Apps for Android, iOS, Web & Desktop*. New York, NY, USA: Apress, 2020.

[4] O. Oisin, "Suricata: An Open Source Next Generation Intrusion Detection System," *Black Hat USA*, Las Vegas, NV, USA, 2010.

[5] Wireshark. [Online]. Available: https://www.wireshark.org

[6] Snort. [Online]. Available: https://www.snort.org

[7] tcpdump. [Online]. Available: https://www.tcpdump.org

[8] S. Alseraye, F. Hashim, and A. M. S. Alghuwainem, "A Survey of Network Traffic Monitoring and Analysis Tools," *IEEE Access*, vol. 5, pp. 15797–15808, 2017.

[9] B. L. Smith and J. Doe, "A Comparison of Network Traffic Analysis Tools," in *Proc. IEEE Int. Conf. Netw. Secur.*, Dubai, UAE, 2020, pp. 1–6.

[10] M. A. Qadeer, A. Iqbal, M. Zahid, and M. R. Siddiqui, "Network Traffic Analysis and Intrusion Detection using Packet Sniffer," in *Proc. 2nd Int. Conf. Commun. Softw. Netw.*, Singapore, 2010, pp. 313–317.

[11] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Secur.*, vol. 45, pp. 100–123, Sep. 2014.

[12] W. Lee and S. J. Stolfo, "Data mining approaches for intrusion detection," in *Proc. 7th USENIX Secur. Symp.*, San Antonio, TX, USA, 1998, pp. 79–94.

[13] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, Secondquarter 2016.

[14] V. Jacobson, C. Leres, and S. McCanne, "Libpcap," Lawrence Berkeley National Laboratory, Berkeley, CA, USA, 1989. [Online]. Available: https://www.tcpdump.org/manpages/pcap.3pcap.html

[15] M. Jones, "Comparing Packet Capture Libraries: libpcap vs. Scapy," *J. Netw. Softw. Tools*, vol. 1, no. 1, pp. 45–52, 2019.

[16] PRTG Network Monitor. [Online]. Available: https://www.paessler.com/prtg

[17] SolarWinds Network Performance Monitor. [Online]. Available: https://www.solarwinds.com/npm

[18] A. Brown and B. Davis, "Web-Based Network Monitoring: Trends and Benefits," *IEEE Netw.*, vol. 36, no. 3, pp. 10–15, May/Jun. 2022.

[19] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, "A survey on the interplay between the Internet of Things and cybersecurity," *ACM Comput. Surv.*, vol. 48, no. 1, pp. 1–36, Jul. 2015.