Списание „Математика, компютърни науки и образование“ Том 9, кн. 1, 2026
Journal of Mathematics, Computer Science and Education Vol. 9, No. 1, 2026

10.54664/YAWE8900

PYTHON BASED NETWORK ANALYSIS ALGORITHM FOR HANDLING RAW PACKET CAPTURES

Vasil Andonov

Abstract: *Modern network analysis places demands on tools that transform raw packet captures into actionable intelligence with minimal manual effort. The algorithm in this paper rises to this challenge through its innovative dual engine approach, combining the efficiency of dpkt for packet parsing with PyShark's depth for DNS analysis. The tool automatically classifies 21 application protocols, maps traffic geolocations via GeoIP2, and distinguishes intranet, VPN, and internet communications—all while generating structured Excel reports. Unlike conventional analyzers, it detects protocols through both port and payload inspection, revealing hidden traffic patterns. This solution reduces the time for analysing the raw data and the focus goes to the already filtered data.*

Keywords: *python, analysis, monitoring, network traffic.*

INTRODUCTION

The rapid evolution of network technologies and the increasing sophistication of cyber threats have made advanced packet analysis indispensable for modern cybersecurity and network management. Traditional tools like *Wireshark*, *Nmap,* or *Tshark* require significant expertise and manual effort to extract meaningful insights from packet captures, particularly when analyzing large datasets or correlating information across multiple dimensions (Wireshark Foundation., 2023), (TShark.dev., 2023), (Nmap Project., 2023). This gap in automated, intelligent analysis motivates the development of more sophisticated solutions that can transform raw network data into actionable intelligence with minimal human intervention.

The algorithm represents a step forward in automated network traffic analysis by combining high performance packet processing with feature extraction. The system employs a dual analysis approach, leveraging the *dpkt* library for efficient packet parsing and *PyShark* for specialized *DNS (Domain Name System)* inspection, ensuring both comprehensive coverage and depth of analysis (PyShark Documentation., 2023), (dpkt Documentation., 2023). Beyond basic protocol identification, the tool implements advanced techniques including payload inspection for protocol detection, enabling it to recognize traffic even when using nonstandard ports. This capability proves particularly valuable for identifying potential security threats that might otherwise evade traditional port-based detection methods.

A key differentiator of this solution lies in its contextual understanding of network traffic. By integrating *GeoIP2* databases, the analyzer enriches raw packet data with geographical and organizational context, mapping *IP* addresses to physical locations and autonomous systems (MaxMind., 2023). This geolocation capability, combined with traffic classification into intranet, *VPN (Virtual Private Network)*, and internet categories, provides security teams with crucial situational awareness for threat investiga-

tion and network monitoring. The system's output goes beyond typical packet analysis tools by automatically generating structured *Excel* reports complete with traffic statistics, protocol distributions, and *DNS* records, significantly reducing the time required for manual report generation.

In the context of network analysis, it is worth mentioning *Arkime*, an open-source tool designed for large scale, full packet capture and analysis *(https://arkime.com/learn)*. While *Arkime* is a more advanced and comprehensive solution, the methodology presented in this article remains valuable. Its approach can be adapted for specialized analysis tasks and, like many open-source tools, can be freely modified to suit specific requirements.

The importance of this work extends across multiple domains, from enterprise security operations to academic research. This paper presents the design principles, implementation details, and evaluation results of the algorithm, demonstrating its effectiveness through real world case studies and discussing future directions for enhancing its capabilities in areas such as real time analysis and encrypted traffic inspection.

EXPOSITION

Algorithm Architecture

The algorithm follows a multi-layered architecture designed (Fig. 1.) to process network traffic data efficiently and accurately. The system begins with packet ingestion, where raw *PCAP (Packet Capture)* files are parsed using a dual engine approach: *dpkt* for high - speed packet decoding and *PyShark* for deep protocol inspection. This initial layer normalizes packet data into a standardized format, preserving critical metadata such as *IP* headers, transport protocols, and payload information.

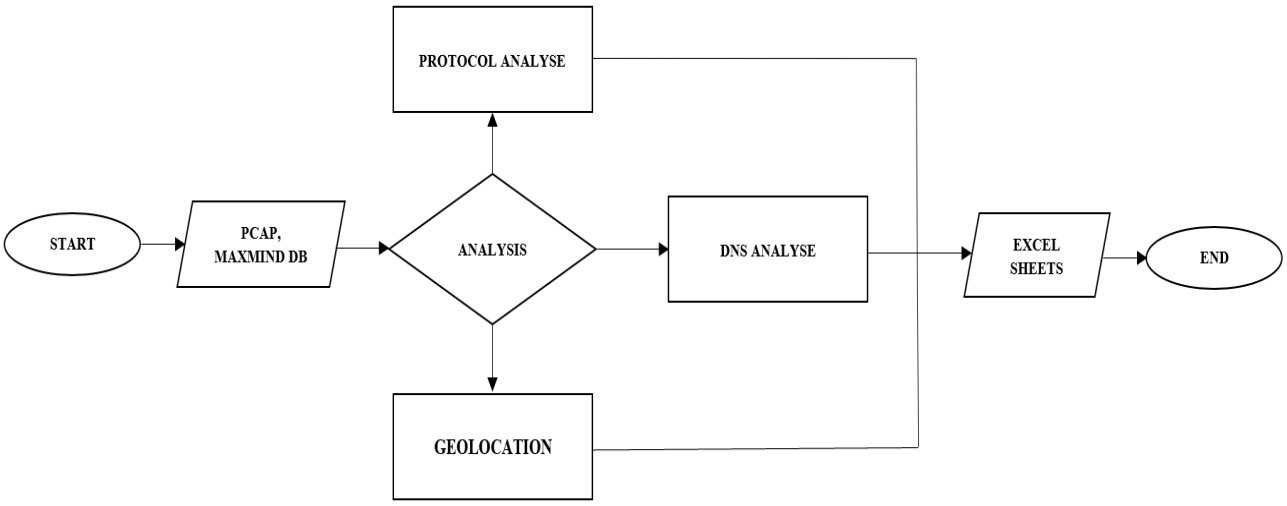

Fig. 1. *Block scheme of the algorithm.*

The protocol analysis layer implements hierarchical classification, starting with transport layer identification *(TCP/UDP ports)* and progressing to application layer detection. The latter combines port-based matching with payload signature analysis to recognize protocols even on nonstandard ports. Specialized handlers process *DNS* traffic separately, extracting queries, responses, and *TTL (Time to live)* values while maintaining request-response relationships (Lopez, M. A., & Moon, S., 2015).

Following protocol analysis, the contextual enrichment layer enhances the data with geolocation and network classification. Using *MaxMind*'s *GeoIP2* databases, *IP* addresses are mapped to countries, *ASNs (Autonomous System Number)*, and organizations. Traffic is categorized as *intranet*, *VPN*, or *internet* based on source and destination *IP* ranges. A caching mechanism optimizes lookup performance for recurring addresses.

The final reporting layer aggregates processed data into structured outputs. Statistical modules generate protocol distributions, traffic flows, and *DNS* records, which are formatted into multi-sheet *Excel* reports.

Protocol Analysis

The Protocol Analysis component forms the core layer of the algorithm, employing a multi- stage classification system to identify and categorize network traffic. The process begins with transport layer inspection, where packets are classified as *TCP* or *UDP* through examination of *IP* header fields. This initial filtering enables optimized processing pathways, with separate analysis routines for connection oriented *(TCP)* and connectionless *(UDP)* traffic. The system maintains protocol state awareness throughout this phase, tracking session initiation, data transfer patterns, and termination signals where applicable, providing crucial context for subsequent application layer analysis (GeeksforGeeks., 2023), (Laurens D'hooge., 2018).

At the application layer detection stage, the analyzer implements a hybrid identification methodology combining both conventional and advanced techniques *(Fig.2.)*. Primary classification occurs through an optimized port matching engine that references an extensive protocol database covering *21* common services *(HTTP, HTTPS, SMTP, SMTPS, POP3, POP3S, IMAP, IMAPS, FTP, FTPS, TFTP, SFTP/SSH, DNS, DHCP - SERVER, DHCP - CLIENT, SNMP, SIP, SIPS, RTP, LDAP, LDAPS),* (Technical Guides & Articles Scaler., 2023). To counter evasion techniques, a secondary payload signature analysis module activates for ambiguous traffic, scanning initial packet bytes for characteristic protocol markers *(Fig.3.)*. This dual approach proves particularly effective in identifying *HTTP* traffic on nonstandard ports, where the presence of header patterns like "*GET/POST"* overrides port-based assumptions. The system further enhances accuracy through statistical analysis of packet timing and size distributions, creating additional validation points for protocol confirmation (Hassan, R., 2020), (Sai Yathin Manugula., 2024).

```
# dpkt for fast parsing + PyShark for deep DNS
with open(pcap_file, 'rb') as f:  # dpkt
    pcap = dpkt.pcap.Reader(f)
cap = pyshark.FileCapture(pcap_file)  # PyShark
```

Fig. 2. *Dpkt and PyShark analysis.*

```
# Port-based + payload checks (e.g., HTTP detection)
if b'HTTP' in tcp.data[:10]:  # Payload check
    return ('HTTP', str(port))
```

Fig. 3. *Port - based and payload check.*

The architecture incorporates continuous learning mechanisms, with an extensible rules engine that allows dynamic updates to protocol signatures and detection. Performance optimization is achieved through selective parsing—applying full deep packet inspection only when preliminary classification indicates ambiguity or potential security relevance. All protocol attributions include confidence scoring, enabling downstream components to appropriately weight analysis results during reporting and alert generation (Kumar, V., 2023). This sophisticated approach to protocol analysis provides the foundation for the system's advanced traffic characterization capabilities while remaining adaptable to evolving network protocols and emerging threats.

Geolocation

The geolocation module serves as a critical component of the algorithm, transforming raw *IP* addresses into meaningful contextual data that enhances network visibility and security analysis. By integrating *MaxMind's GeoIP2* databases, the system provides comprehensive geographical and organizational attribution for all public *IP* addresses encountered in network traffic. The module first performs *IP* classification, distinguishing between private *RFC 1918* addresses and public internet addresses, with only the latter undergoing full geolocation resolution. This selective processing optimizes performance while maintaining complete network context.

For each public *IP* address, the analyzer retrieves multiple layers of contextual information through efficient database lookups. The *GeoLite2-City* database provides country level and, when available, city level geographical attribution, while the *GeoLite2-ASN* database supplies *ASN* and *ISP (internet service provider)* information. The system implements a smart caching mechanism that stores recent lookup results, significantly reducing processing time for recurring *IP* addresses commonly found in network traffic, such as those belonging to cloud providers, content delivery networks, or frequently accessed services. This caching strategy maintains analysis speed even when processing large packet captures containing thousands of unique *IP* addresses **(**MaxMind.**,** 2023).

Beyond basic geolocation, the module performs traffic classification by analyzing *IP* address relationships. It automatically categorizes network flows into three distinct types: intranet traffic between private *IP* addresses, *VPN* traffic between private and public *IPs*, and standard internet traffic between public *IPs*. This classification proves particularly valuable for security monitoring, as it helps identify potential policy violations, such as unauthorized external communications from internal hosts or unexpected *VPN* usage patterns. The system further enhances security analysis by flagging connections to high risk geographical regions or suspicious autonomous systems based on configurable threat intelligence feeds (Patel, S., 2022).

The geolocation data integrates seamlessly with the analyzer’s reporting framework, appearing in both detailed traffic logs and summary visualizations. In the *Excel* output, each flow record includes fields for source and destination countries, *ASN* and organization names, enabling quick filtering and sorting by geographical attributes (Cloudflare., 2023). The summary sheet provides aggregated views of traffic by country and *ASN*, helping analysts identify dominant geographical patterns or unexpected network relationships at a glance. This implementation strikes an optimal balance between depth of analysis and processing efficiency, making it suitable for both routine network monitoring and in-depth forensic investigations (Andersson, L., 2022).

DNS Analysis

The *DNS* analysis module provides specialized processing of *Domain Name System* traffic, offering critical insights for both network troubleshooting and security investigations. The system captures and analyzes *DNS* packets through a multistage approach that begins with protocol identification, where it distinguishes between standard *DNS (UDP/53)* and *DNS over TCP* traffic. For each *DNS* transaction, the analyzer extracts and correlates query/response pairs using transaction *IDs*, ensuring proper session tracking even in high volume traffic environments. This correlation capability proves particularly valuable when investigating potential *DNS* tunnelling or exfiltration attempts, as it maintains the complete context of name resolution activities (Research Papers & Whitepapers Cisco., 2020), (TechTarget., 2021).

At the core of the *DNS* analysis lies the comprehensive parsing of *DNS* resource records. The system decodes *A* and *CNAME* record types. The module implements *TTL* analysis in strict accordance with *DNS* standards, interpreting values in seconds and tracking *TTL* patterns that could suggest fast flux networks or other evasion techniques.

All *DNS* analysis results integrate seamlessly with the system’s reporting framework. The dedicated *DNS* worksheet in the *Excel* output presents both raw *DNS* transaction data and analyzed metrics, including comprehensive lists of queried domains with their corresponding responses, response times, and *TTL* values.

Experiment

The experimental analysis is based on *54MB PCAP* file with *132,217* packets in it. The time needed for the algorithm to handle it is *36 sec* and the output file *(Excel)* is *5.85 MB - Intel (R) Core (TM) i5 - 8265U CPU @ 1.60GHz - 1.80 GHz* processor and *16 GB RAM*. To further evaluate performance, the algorithm was tested on ten additional *PCAP* files of varying sizes *(23 MB - 174 MB)* and packet counts *(53,348–322,576)*. The experiments confirmed that all packets were successfully processed, with the only variation being the total *handling time* of each file.

1	**PCAP File Analysis Summary**	
2		
3	**Basic Statistics**	
4	Total Packets	132217
5	IPv4 Packets	76856
6	IPv6 Packets	55361
7	Unique IP Addresses	0
8	Error Packets	0
9		
10	**Protocol Distribution**	
11	UDP	70655
12	TCP	61510
13	HTTPS	21135
14	DNS	5110
15	ICMPv6	52
16	RTP	4
17		
18	**Network Analysis**	
19	Most Common Source Countries	Not Available, Bulgaria, Private Network
20	Most Common Destination Countries	Private Network, Bulgaria, Not Available
21	Most Common Source Organizations	CLOUDFLARENET, Vivacom Bulgaria EAD, Private Network
22	Most Common Destination Organizations	Private Network, Vivacom Bulgaria EAD, CLOUDFLARENET
23		
24	**DNS Analysis**	
25	Total DNS Responses	600

Fig. 4. *PCAP Analysis Summary sheet – 1 sheet.*

The *PCAP* analysis summary *(First Sheet)* presents an overview of network traffic captured during a monitoring session, shedding light on protocol usage, communication sources and destinations, and *DNS* behaviour *(Fig. 4.)*. Out of a total of *132,217* packets, there was a significant presence of both *IPv4 (76,856 packets)* and *IPv6 (55,361 packets)* traffic, highlighting either a transitional or dual stack network configuration. Protocol wise, the majority of packets were split between *UDP (70,655 packets)* and *TCP (61,510 packets)* which is typical of networks that support a blend of high speed, low latency services, along with more reliable communications. *HTTPS* traffic totalled *21,135* packets, underscoring the emphasis on secure web interactions, while *DNS (5,110 packets)* and a minimal number of *RTP (4 packets)* and *ICMPv6 (52 packets)* packets suggested modest levels of real time communication and diagnostic activities. Notably, there were no error packets captured, reflecting a clean, loss-free network environment.

From a geographical and organizational perspective, source and destination data pointed to a mix of traffic involving *Bulgaria*, *private networks*, and international services such as *Cloudflare*— an indication of both local *ISP* utilization and secure, *CDN*-based content delivery. The relatively low number of *DNS* responses *(600)* in contrast to the query volume may indicate the use of caching mechanisms or packet loss at specific capture points. Overall, the data suggests a functional, actively used network supporting both internal and external services, with a strong lean towards modern protocol adoption and secure traffic handling.

The table *(Second Sheet)* displays detailed network flow data with each row representing a communication event between two endpoints, identified by their source and destination IP addresses *(Fig. 5.)*. Various attributes are recorded for each flow, including transport and application layer protocols, port numbers, packet size, and metadata regarding geographic and organizational origins. The application protocols primarily include *HTTPS* and *DNS*, suggesting a strong focus on secure web access and domain resolution. Cloudflare appears frequently as both a source and destination organization, reflecting traffic that likely involves content delivery or protection services. Transport protocols are dominated by *TCP* and *UDP*, consistent with the identified applications. Packet sizes range across entries, indicating both lightweight requests and more data-heavy responses.

src_ip	dst_ip	transport_proto	app_proto	app_port	packet_size	src_country	src_asn	src_org	dst_country	dst_asn	dst_org	network_type
2a01:5a8:2:9::d427:441b	2a01:5a8:421:7884:e1bb:fb9d:716e:fdae	TCP	Port 60751	60751	86	Bulgaria	AS8866	Vivacom Bulgaria EAD	Bulgaria	AS8866	Vivacom Bulgaria EAD	Internet
2a04:4e42:70::367	2a01:5a8:421:7884:e1bb:fb9d:716e:fdae	TCP	Port 60741	60741	86	Bulgaria	AS54113	FASTLY	Bulgaria	AS8866	Vivacom Bulgaria EAD	Internet
172.20.10.4	212.39.68.17	TCP	HTTPS	443	55	Private Network	Private Networ	Private Network	Bulgaria	AS8866	Vivacom Bulgaria EAD	VPN
172.20.10.4	146.75.1.91	TCP	HTTPS	443	55	Private Network	Private Networ	Private Network	Bulgaria	AS54113	FASTLY	VPN
172.20.10.4	18.165.72.33	TCP	HTTPS	443	55	Private Network	Private Networ	Private Network	United States	AS16509	AMAZON-02	VPN
212.39.68.17	172.20.10.4	TCP	Port 60755	60755	66	Bulgaria	AS8866	Vivacom Bulgaria EAD	Private Network	Private Network	Private Network	VPN
18.165.72.33	172.20.10.4	TCP	Port 60756	60756	66	United States	AS16509	AMAZON-02	Private Network	Private Network	Private Network	VPN
146.75.1.91	172.20.10.4	TCP	Port 60758	60758	66	Bulgaria	AS54113	FASTLY	Private Network	Private Network	Private Network	VPN
172.20.10.4	212.39.68.42	TCP	HTTPS	443	55	Private Network	Private Networ	Private Network	Bulgaria	AS8866	Vivacom Bulgaria EAD	VPN
212.39.68.42	172.20.10.4	TCP	Port 60757	60757	66	Bulgaria	AS8866	Vivacom Bulgaria EAD	Private Network	Private Network	Private Network	VPN
2a01:5a8:421:7884:e1bb:fb9d:716e:fdae	2606:4700::6812:ed8	TCP	HTTPS	443	75	Bulgaria	AS8866	Vivacom Bulgaria EAD	Not Available	AS13335	CLOUDFLARENET	Internet
2606:4700::6812:ed8	2a01:5a8:421:7884:e1bb:fb9d:716e:fdae	TCP	Port 60740	60740	86	Not Available	AS13335	CLOUDFLARENET	Bulgaria	AS8866	Vivacom Bulgaria EAD	Internet
2a01:5a8:421:7884:e1bb:fb9d:716e:fdae	2a04:4e42:70::347	TCP	HTTPS	443	75	Bulgaria	AS8866	Vivacom Bulgaria EAD	Bulgaria	AS54113	FASTLY	Internet
2a04:4e42:70::347	2a01:5a8:421:7884:e1bb:fb9d:716e:fdae	TCP	Port 60759	60759	86	Bulgaria	AS54113	FASTLY	Bulgaria	AS8866	Vivacom Bulgaria EAD	Internet
2a01:5a8:421:7884:e1bb:fb9d:716e:fdae	2606:4700:7000::6715:f408	TCP	HTTPS	443	86	Bulgaria	AS8866	Vivacom Bulgaria EAD	Not Available	AS13335	CLOUDFLARENET	Internet
2a01:5a8:421:7884:e1bb:fb9d:716e:fdae	2606:4700::6812:572a	TCP	HTTPS	443	75	Bulgaria	AS8866	Vivacom Bulgaria EAD	Not Available	AS13335	CLOUDFLARENET	Internet
2606:4700::6812:572a	2a01:5a8:421:7884:e1bb:fb9d:716e:fdae	TCP	Port 60518	60518	86	Not Available	AS13335	CLOUDFLARENET	Bulgaria	AS8866	Vivacom Bulgaria EAD	Internet
2a01:5a8:421:7884:e1bb:fb9d:716e:fdae	2606:4700:60:0:186:b4f3:7124:2020	TCP	HTTPS	443	75	Bulgaria	AS8866	Vivacom Bulgaria EAD	Not Available	AS13335	CLOUDFLARENET	Internet
2a01:5a8:421:7884:e1bb:fb9d:716e:fdae	2606:4700::6812:572a	TCP	HTTPS	443	75	Bulgaria	AS8866	Vivacom Bulgaria EAD	Not Available	AS13335	CLOUDFLARENET	Internet
2606:4700:60:0:186:b4f3:7124:2020	2a01:5a8:421:7884:e1bb:fb9d:716e:fdae	TCP	Port 60260	60260	86	Not Available	AS13335	CLOUDFLARENET	Bulgaria	AS8866	Vivacom Bulgaria EAD	Internet
2606:4700::6812:572a	2a01:5a8:421:7884:e1bb:fb9d:716e:fdae	TCP	Port 60532	60532	86	Not Available	AS13335	CLOUDFLARENET	Bulgaria	AS8866	Vivacom Bulgaria EAD	Internet
2a01:5a8:421:7884:e1bb:fb9d:716e:fdae	2606:4700:7000::6715:f408	TCP	HTTPS	443	86	Bulgaria	AS8866	Vivacom Bulgaria EAD	Not Available	AS13335	CLOUDFLARENET	Internet
172.20.10.4	103.21.244.9	TCP	HTTPS	443	66	Private Network	Private Networ	Private Network	Not Available	AS13335	CLOUDFLARENET	VPN

Fig. 5. *Traffic Analysis sheet – 2 sheet.*

The geographic distribution involves traffic sourced from countries like *Bulgaria* and the *United States*, with many entries also marked as originating from private network spaces. The *ASNs* and associated organizations, such as *CLOUDFLARENET* and *Vivacom Bulgaria*, highlight key infrastructure providers routing this traffic. Interestingly, the majority of the traffic is categorized as belonging to a *“public”* network type, suggesting open internet access rather than traffic confined to a corporate or internal network.

This table *(Third Sheet)* provides insight into *DNS* resolution activity within a private network, capturing a range of domain queries and their corresponding responses *(Fig. 6.)*. Each row represents a *DNS* lookup event, showing queries for well-known domains such as edge.microsoft.com, static.edge.microsoftapp.net, graph.microsoft.com, th.bing.com, and various *Brave*- and *Google*-associated *URLs*. The repeated presence of *CNAME (Canonical Name)* records indicates that these domains often redirect or alias to more specific service endpoints such as *Edge CDN* hosts or *Microsoft’s* identity services before resolving to an *A (Address)* record with an *IP* address.

	Query	Type	Answer	TTL	Source IP	Destination IP
1						
2	static.edge.microsoftapp.net	CNAME	edge-cloud-resource-static.azureedge.net	3285	172.20.10.1	172.20.10.4
3	static.edge.microsoftapp.net	A	13.107.246.45	3285	172.20.10.1	172.20.10.4
4	static.edge.microsoftapp.net	CNAME	edge-cloud-resource-static.azureedge.net	3285	172.20.10.1	172.20.10.4
5	edge.microsoft.com	CNAME	edge-microsoft-com.ax-0002.ax-msedge.net	2032	172.20.10.1	172.20.10.4
6	edge.microsoft.com	CNAME	edge-microsoft-com.ax-0002.ax-msedge.net	2032	172.20.10.1	172.20.10.4
7	edge.microsoft.com	A	150.171.27.11	2032	172.20.10.1	172.20.10.4
8	edge.microsoft.com	CNAME	edge-microsoft-com.ax-0002.ax-msedge.net	2029	172.20.10.1	172.20.10.4
9	edge.microsoft.com	A	150.171.27.11	2029	172.20.10.1	172.20.10.4
10	edge.microsoft.com	CNAME	edge-microsoft-com.ax-0002.ax-msedge.net	2029	172.20.10.1	172.20.10.4
11	th.bing.com	CNAME	p-th.bing.com.trafficmanager.net	4433	172.20.10.1	172.20.10.4
12	quorum.wdp.brave.com	CNAME	safe-browsing-quorum.wdp-public.brave.com	377	172.20.10.1	172.20.10.4
13	ecn.dev.virtualearth.net	CNAME	ssl2.tiles.virtualearth.net.edgekey.net	3243	172.20.10.1	172.20.10.4
14	graph.microsoft.com	CNAME	ags.privatelink.msidentity.com	1966	172.20.10.1	172.20.10.4
15	deepseek-user-avatar.obs.cn-east-3.myhuaweicloud.com	A	121.36.239.137	152	172.20.10.1	172.20.10.4
16	brave-today-cdn.brave.com	CNAME	d18pg1ten86q1d.cloudfront.net	342	172.20.10.1	172.20.10.4
17	edge.microsoft.com	CNAME	edge-microsoft-com.ax-0002.ax-msedge.net	1979	172.20.10.1	172.20.10.4
18	edge.microsoft.com	A	150.171.27.11	1979	172.20.10.1	172.20.10.4
19	edge.microsoft.com	CNAME	edge-microsoft-com.ax-0002.ax-msedge.net	1979	172.20.10.1	172.20.10.4
20	lh3.googleusercontent.com	CNAME	googlehosted.l.googleusercontent.com	46	172.20.10.1	172.20.10.4
21	static01.nyt.com	CNAME	static.prd.map.nytimes.com	625	172.20.10.1	172.20.10.4
22	edge.microsoft.com	CNAME	edge-microsoft-com.ax-0002.ax-msedge.net	1912	172.20.10.1	172.20.10.4
23	edge.microsoft.com	A	150.171.27.11	1912	172.20.10.1	172.20.10.4

Fig. 6. *DNS Analysis sheet – 3 sheet.*

The *TTL* values, ranging from under a minute to over an hour, suggest a mix of transient and more persistent cache strategies. Every entry originates from the same internal *IP (172.20.10.1)* and targets the same destination *(172.20.10.4)*, pointing to a likely *client–server* relationship in a *local network*. These *DNS* requests align with activity like web browsing, app updates, and telemetry reporting, especially interactions with *Microsoft* services.

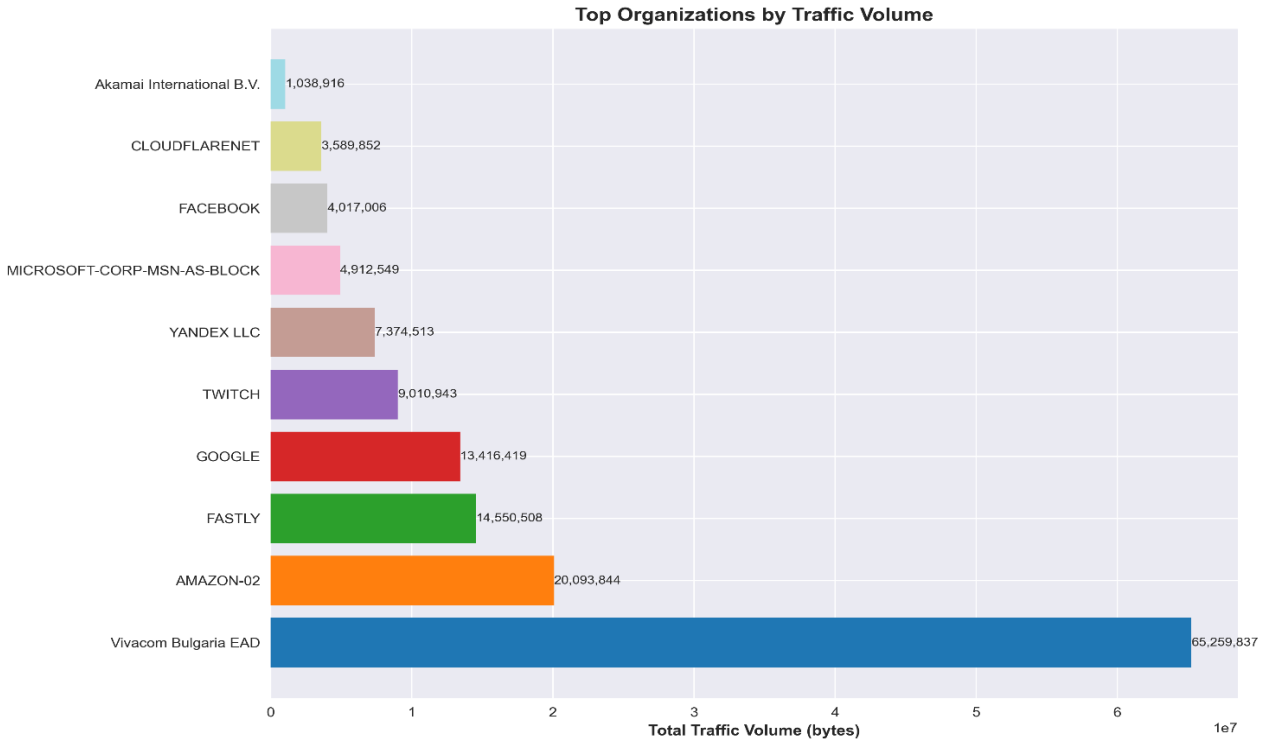

Fig. 7. *Top Organizations by Traffic Volume*

As a final output of the analysis *(Fig. 7, 8)*, the algorithm uses *Matplotlib* to generate comprehensive dashboards summarizing the key insights from the *PCAP* file. These *dashboards* provide a clear overview of protocol and network type distributions, top countries by traffic volume, and leading organizations. This visual representation is significantly easier to interpret than raw Excel sheets and provides a ready foundation for more in depth analysis.

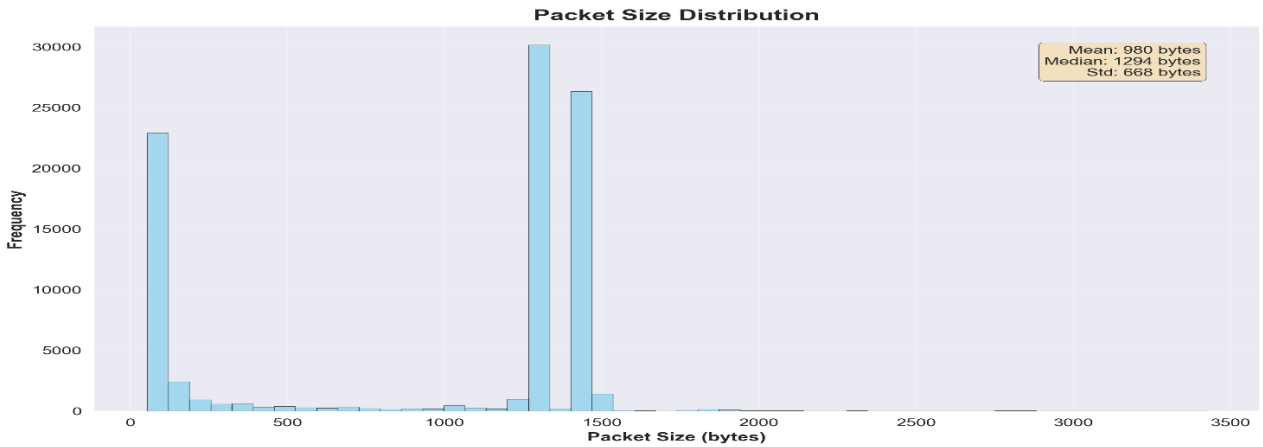

Fig. 8. *Geographic Distribution - Top Countries and Packet Size Distribution*

CONCLUSION

The algorithm represents a significant advancement in network traffic analysis, addressing critical gaps in existing tools by automating the transformation of raw packet data into structured, actionable intelligence. Through its innovative dual engine architecture combining *dpkt* for high-speed parsing and *PyShark* for deep protocol inspection, the system delivers comprehensive visibility into network communications while maintaining processing efficiency. This approach enables security teams and network administrators to bypass the tedious manual analysis typically required with traditional tools, instead focusing their efforts on interpreting meaningful insights.

A key strength of the solution lies in its multi-layered analysis capabilities. The tool's sophisticated protocol detection goes beyond simple port matching to incorporate payload inspection, revealing evasive traffic patterns that would otherwise remain hidden. The integration of *GeoIP2* databases provides crucial contextual awareness by mapping *IP* addresses to geographical locations and organizational entities, while traffic classification distinguishes between *intranet*, *VPN*, and *internet* communications. The specialized *DNS* analysis module offers particular value for security investigations, maintaining complete query/response context and detecting potential anomalies through *TTL* analysis and domain pattern recognition.

The system's practical impact is most evident in its automated reporting functionality, which transforms analyzed data into professional grade *Excel* outputs complete with traffic statistics, protocol distributions, and *DNS* records. This feature alone represents a substantial time savings over manual report generation, making the tool valuable for both routine monitoring and in-depth forensic investigations.

This algorithm establishes a foundation for a full-scale network analysis application. Future development paths include enhancing its capabilities to detect session time anomalies, identify network overload indicative of *DDoS* attacks, and analyze packets for malware. Its modular, open-source design makes it well suited for such expansions.

REFERENCES

1. Arkime. Network Analysis & Packet Capture. 2026. URL: https://arkime.com/learn.

2. Andersson, L. (2022). DNS for Cybersecurity. DIVA Portal. URL: https://www.diva-portal.org/smash/get/diva2:1873919/FULLTEXT01.pdf

3. Cloudflare. (2023). What is an Autonomous System (AS)? URL: https://www.cloudflare.com/learning/network-layer/what-is-an-autonomous-system/

4. Cloudflare. (2023). What is DNS? URL: https://www.cloudflare.com/learning/dns/what-is-dns/

5. Cisco. (2020). DNS Configuration Guide. URL: https://www.cisco.com/c/en/us/td/docs/net_mgmt/prime/network_registrar/111/dns/guide/DNS_Guide/DNS_Guide_chapter_00.pdf

6. dpkt Documentation. (2023). Python Packet Parsing Library. URL: https://dpkt.readthedocs.io/en/latest/

7. GeeksforGeeks. (2023). Transport Layer Protocols. URL: https://www.geeksforgeeks.org/computer-networks/transport-layer-protocols/

8. Hassan, R. (2020). Threat Hunting with Network Data. SANOG36 Tutorial. URL: https://sanog.org/resources/sanog36/SANOG36-Tutorial_ThreatHunting_Hassan.pdf

9. Kumar, V. (2023). Real-Time Network Monitoring. IARJSET ICMART-24. URL: https://iarjset.com/wp-content/uploads/2023/06/IARJSET-ICMART-24.pdf

10. Laurens D'hooge. (2018). Network traffic profiling and anomaly detection for cyber security. URL: https://libstore.ugent.be/fulltxt/RUG01/002/494/830/RUG01-002494830_2018_0001_AC.pdf

11. Lopez, M. A., & Moon, S. (2015). Machine Learning for Network Traffic Analysis. arXiv:1507.05722. URL: https://arxiv.org/pdf/1507.05722

12. MaxMind. (2023). GeoIP2 Geolocation Databases. URL: https://www.maxmind.com/en/home;

13. Nmap Project. (2023). Nmap: Free Security Scanner. URL: https://nmap.org/

14. Patel, S. (2022). Advanced PCAP Analysis Techniques. IJRPR, 4(11), 12–25. URL: https://ijrpr.com/uploads/V4ISSUE11/IJRPR19300.pdf

15. PyShark Documentation. (2023). Python Wrapper for TShark. URL: https://pyshark-packet-analysis.readthedocs.io/en/latest/

16. Sai Yathin Manugula, Dheeraj Varma Kalidindi, Sindhu Sri Gogikari and Srinivas Rao Billakanti. (2025). Anomaly detection in network traffic using azure machine learning and log analytics. *World Journal of Advanced Research and Reviews*. URL: https://journalwjarr.com/sites/default/files/fulltext_pdf/WJARR-2025-2197.pdf

17. Technical Guides & Articles Scaler. (2023). Application Layer Protocols in Computer Networks. URL: https://www.scaler.com/topics/computer-network/application-layer-protocols/

18. TechTarget. (2021). Network Packet Analysis Fundamentals. URL: https://cdn.ttgtmedia.com/rms/pdf/MPNS_CH5.pdf

19. TShark.dev. (2023). TShark Command-Line Network Analyzer. URL: https://tshark.dev/;

20. Wireshark Foundation. (2023). Wireshark: Network Protocol Analyzer. URL: https://www.wireshark.org/

ABOUT THE AUTHOR

Vasil Andonov, cadet, Faculty of Security and Defense, Vasil Levski National Military University, Bulgaria, E-mail: vasilandonov123@gmail.com