

Journal Scientific and Applied Research, Vol. 30, 2026
International Journal

ISSN 1314-6289 (Print)
ISSN 2815-4622 (Online)

Original Contribution

PASSIVE METHODS FOR NETWORK FORENSICS AND DEEP PACKET INSPECTION (DPI) IN LOCAL AREA NETWORKS

Denis I. Zvezdelinov, Zdravko Y. Kuzmanov[2]

[1]Department of Communication and Computer Engineering and Security Technologies, Faculty of Technical Sciences, Konstantin Preslavsky University of Shumen, Shumen 9712, 115 Universitetska St., e-mail: d.zvezdelinov@shu.bg
[2]Department of Communication and Computer Engineering and Security Technologies, Konstantin Preslavsky University of Shumen, Shumen 9712, 115 Universitetska St., e-mail: z.kuzmanov@shu.bg

Abstract: *Network forensics is a critical element of the modern cybersecurity incident investigation process. Unlike active scanning methods, passive analysis enables the extraction of evidentiary material without modifying network traffic and without the risk of exposing the analysis station. This report examines the application of passive Deep Packet Inspection (DPI) methods within Local Area Network (LAN) environments. The focus is placed on the technological framework for Full Packet Capture (FPC), the chronological reassembly of TCP/IP sessions, and the decoding of transport and application-layer protocol structures. By utilizing open-source software tools (tcpdump, Wireshark, TShark), the full cycle of forensic analysis is demonstrated – ranging from the generation of raw .pcap datasets to the extraction of Indicators of Compromise (IoCs) and the reconstruction of transferred files. The report outlines key methodological recommendations for preserving the integrity of digital evidence and analyzes the challenges arising from the widespread adoption of encrypted protocols (TLS/HTTPS).*

Key words: *Network forensics, Deep Packet Inspection (DPI), Full Packet Capture, Pcap, Wireshark, Tcpdump, Reassembly of sessions.*

1. Introduction

In modern information infrastructure, digital forensics has established itself as a key discipline for the detection, analysis, and documentation of computer incidents. Within this scientific and applied domain, network forensics occupies a distinct position, focusing on the capture, recording, and analysis of network events to identify the sources of cyberattacks, unauthorized access, or the leakage of sensitive information. Unlike traditional host-based forensics (Host/Disk Forensics), which investigates static artifacts on physical media (file systems, registries, operating system logs, or random-access memory), network forensics operates with dynamic data in motion (data-in-motion). Once network traffic passes through a given communication channel, it is irretrievably lost unless it has been intentionally captured

and stored in real time. This makes the precision and methodology of the capture crucial for the entire subsequent investigation [2].

A critical requirement when conducting network analysis is the implementation of passive monitoring methods. Any active operations (such as reverse scanning, sending probing packets, or traffic injection) carry a significant risk of altering the state of the investigated system, triggering the attacker's defense mechanisms (IDS/IPS), or even destroying digital evidence. Passive analysis strictly adheres to the principle of non-intrusiveness:

- Stealth Mode: The analytical station operates entirely as a passive observer, without generating any traffic of its own or revealing its presence on the network.
- Preservation of Evidence Integrity: It guarantees that the original data stream remains unmodified, which is a mandatory condition for maintaining the legal and operational validity of the collected evidence (Chain of Custody).

The present report aims to systematize and demonstrate practical methodologies for passive network analysis and Deep Packet Inspection (DPI) within local area networks (LAN). It explores the complete technological cycle – from low-level raw packet capture (Full Packet Capture) using specialized open-source software (tcpdump) [7], through the chronological recomposition of TCP sessions and the dissection of application protocols (Wireshark/TShark) [10], to the extraction of Indicators of Compromise (IoCs) and the analysis of challenges posed by modern encrypted traffic [1].

2. Conceptual framework for data capture and evidence validation

To conduct a successful digital investigation at the network level, it is necessary to apply a strict methodological framework that regulates the collection, validation, and storage of network data. Imprecise collection or compromising the integrity of raw files can lead to erroneous conclusions or the complete invalidation of the collected digital evidence. In network monitoring practice, there are two main approaches to gathering data from communication channels: collecting flow metadata (Flow Data) and Full Packet Capture (FPC). Flow metadata (NetFlow / IPFIX / sFlow) records only statistical data about the communication session – source and destination IP addresses, source and destination ports, transport protocol used, number of transferred packets/bytes, and session duration. Flow data provides an excellent overview of the network load and topological connections ("who talked to whom"), requiring a minimal amount of disk space. The FPC model stores an absolute copy of every transmitted packet, including the headers from all OSI model layers and the entire payload [2].

Table 1. Comparison between flow data and full packet capture

Criteria	Flow Data (NetFlow/IPFIX)	Full Packet Capture (FPC)
Data scope	Metadata only (Layer 3/Layer4)	Full packet (Layer 1 – Layer 7)
Resource requirements	Low (minimal disk space)	High (large storage volume)
File reconstruction	Impossible	Fully possible
Deep packet inspection	Not supported	Fully supported
Value	Useful for anomaly detection	Gold standard for investigation

For the purposes of network forensics, Full Packet Capture is recognized as the gold standard. Only FPC allows the investigative team to perform Deep Packet Inspection (DPI), decrypt sessions (if keys are available), reconstruct transferred files, and analyze the content

of malicious exploits. The process of collecting digital evidence requires strict adherence to the principle of traceability and preventing any modifications to the collected data (Chain of Custody). Network traffic is stored in standard formats such as .pcap (Packet Capture) or its modern successor .pcapng (PCAP Next Generation). To guarantee the legal and operational authenticity of the captured files, immediately after stopping the capture process, a cryptographic hash (e.g., SHA-256 or MD5) is calculated on the generated PCAP file. This value serves as a "digital fingerprint". Every subsequent analysis is performed solely on a working copy of the file, while the original is stored encrypted. In case of any doubt regarding tampering, the hash sum is recalculated and compared with the initial one. Unlike the classic .pcap format, .pcapng allows the encapsulation of additional metadata within the file itself – detailed information about the physical interface, the equipment used, the operating system of the analytical node, as well as digital comments entered by the analyst, without altering the raw traffic [6].

The increasing capability of analytical systems to extract and correlate sensitive information also introduces privacy-related risks, particularly when captured datasets are processed by automated or AI-assisted analytical tools. Therefore, forensic effectiveness should be accompanied by appropriate access control, data minimization, and controlled processing mechanisms [9].

The accuracy of timestamps is a critical factor when conducting chronological analysis. When investigating complex cyber incidents, it is necessary to correlate events from multiple sources – network packet captures, operating system logs (Syslog), firewall logs, and databases. When capturing packets at the network interface card/driver level, the operating system assigns a timestamp to each packet. Analyzing high-speed networks and precisely recomposing TCP sessions requires microsecond or nanosecond resolution. The analytical station must be precisely synchronized with a reliable time source via NTP (Network Time Protocol) or PTP (Precision Time Protocol / IEEE 1588). A clock drift of even a few milliseconds can lead to the incorrect ordering of events in the timeline (Timeline Reconstruction) and the inability to determine the actual cause-and-effect relationship of a given attack [2].

3. Principles of deep packet inspection (DPI)

Raw network data capture (Full Packet Capture) is only the first phase of the digital investigation process. To extract actionable security intelligence, the application of Deep Packet Inspection (DPI) is required. Unlike standard network monitoring, DPI enables the analysis of communication content at the application layer of the OSI model. In classic network diagnostics and traditional stateful inspection firewalls, so-called Shallow Packet Inspection is applied. With this approach, analysis is strictly limited to Layer 3 (network layer) and Layer 4 (transport layer) headers – source and destination IP addresses, source and destination TCP/UDP ports, checksums, and TCP flags (SYN, ACK, FIN, RST). Shallow inspection blindly assumes that traffic on port 80 is HTTP and on port 443 is HTTPS. This makes the method easily bypassable by malware that intentionally uses standard ports to tunnel encrypted Command and Control (C2) channels or perform covert data exfiltration.

Deep Packet Inspection (DPI) builds upon this process by scanning lower-layer metadata and inspecting the payload from Layer 5 to Layer 7 (application layer). Through DPI, the analytical system can recognize the actual application protocol, identify the specific software client (via User-Agent headers), extract the contents of transmitted files, and detect specific signatures of known exploits.

Modern network forensics does not rely on port numbers to determine traffic type; instead, it utilizes Port-Independent Protocol Identification (PIPI) methods. DPI engines apply three primary analysis techniques:

- Pattern/Signature Matching: Searching for specific byte sequences, text strings, or regular expressions (Regex) within the payload.
  - Example: The presence of the string HTTP/1.1 200 OK or GET /index.html unambiguously identifies the traffic as HTTP, even if communication occurs on a non-standard port (e.g., port 8080 or 4444).
  - TLS Identification: In encrypted traffic, the byte structure of the Client Hello field during the TLS handshake contains information about the requested domain (via the Server Name Indication – SNI extension), allowing traffic categorization without decryption.

- Protocol Parsing/Decoding: Specialized analyzers (such as dissectors in Wireshark or TShark) simulate the behavior of end applications. They sequentially decode the packet structure according to the respective RFC specifications, dissecting it into individual semantic fields (e.g., DNS queries/responses, SMB file operations, SMTP commands).
- Heuristic & Statistical Analysis: Applied when clear signatures are lacking or with protocols designed to bypass firewalls (e.g., P2P networks). Statistical characteristics such as packet size distribution, inter-arrival times, and flow directions are analyzed. Modern encrypted transport protocols such as QUIC further complicate passive network analysis by combining transport functionality with cryptographic protection over UDP, thereby limiting the visibility available to traditional packet-inspection techniques [4].

Network forensic analysis of single packets is insufficient because application content (e.g., an executable file, image, or document) is split into dozens or hundreds of fragments during transmission. To perform precise DPI, the analytical system must execute stream reassembly at two levels:

- Layer 3 Defragmentation: If an IP packet's size exceeds the Maximum Transmission Unit (MTU), it is split into fragments. Threat actors sometimes employ overlapping IP fragments or intentionally modified fragment offsets to bypass DPI environments. The analytical system must correctly reconstruct the original IP datagram before passing it to upper layers.
- Layer 4 Stream Reassembly: Recomposing TCP sessions for a connection-oriented protocol that guarantees delivery, but not packet ordering during transit. The analytical module maintains a State Table and performs the following tasks:
  - Reordering by Sequence Numbers: Handling out-of-order packets.
  - Duplicate Removal: Eliminating TCP retransmissions caused by network packet loss to avoid duplicating the extracted payload.
  - Handling Delays and Asymmetric Routing: Tracking receive windows (TCP Window) and correctly closing sessions via FIN or RST packets.

Only after the TCP stream is fully reconstructed and normalized (Traffic Normalization) is the payload passed to the analyzers to extract the final artifact content (e.g., reconstructing a malicious .exe file transferred over the network) [6].

4. Software tools for forensic analysis

The effective implementation of Deep Packet Inspection (DPI) methods and Full Packet Capture (FPC) of network traffic requires reliable software. In the field of network forensics, the industry standard is based on open-source solutions that ensure maximum transparency of

dissection algorithms and flexibility in process automation. To realize the full digital investigation cycle – from raw recording to artifact extraction – a combination of Tcpdump, Wireshark, and TShark is utilized. Tcpdump is a command-line network analyzer distinguished by extremely low consumption of system resources (CPU and RAM). This makes it the ideal tool for deployment on analytical stations with limited capacity (such as Raspberry Pi) or for prolonged monitoring in headless environments (servers without a graphical user interface). When conducting forensic recording over long periods of time, a primary challenge is preventing system buffer overflows (which lead to packet drops) and disk space exhaustion. To address these issues, tcpdump offers optimization mechanisms:

- Increasing the Capture Buffer: Via the -B parameter (in kilobytes), the operating system allocates a larger Ring Buffer for the network interface, ensuring that during sudden traffic spikes, packets will be held until the software writes them to disk.
- File Rotation: Instead of creating a single giant .pcap file that is difficult to process later, automatic splitting is used. The -C parameter defines the maximum size of a single file (in megabytes), while -W specifies the maximum number of files to retain before the oldest one is overwritten.
- Capture Filtering: tcpdump utilizes the Berkeley Packet Filter (BPF) syntax, which compiles directly in the operating system kernel for maximum speed. Applying filters such as not port 22 is critical to exclude legitimate remote management traffic (SSH) of the analytical station itself from the capture.

While Tcpdump is optimal for capture, Wireshark is the de facto standard for subsequent (offline) graphical analysis of collected .pcap datasets. Wireshark's strength lies in its built-in dissectors (over 3,000 supported protocols), which automatically decode complex protocol hierarchies and present them in a human-readable format. An analyst's primary tool in Wireshark is Display Filters. Unlike BPF capture filters, display filters do not discard data; they merely hide packets that do not match the specified condition. This allows investigators to flexibly "focus" on suspicious activity:

- Detecting potential data exfiltration: http.request.method == "POST"
- Identifying port scanning (SYN Scan): tcp.flags.syn == 1 and tcp.flags.ack == 0
- Isolating a specific file transfer: smb2.filename contains "secret"

One of the tool's most powerful forensic features is Follow TCP Stream – an option that automatically recomposes the fragmented payload from multiple related packets and renders it as a unified, continuous text or hexadecimal stream. Wireshark's graphical interface becomes inefficient and unstable when attempting to load massive files (for example, over 1 GB). For processing large-scale data volumes, Tshark – the terminal equivalent of Wireshark – is used. TShark allows the use of the same powerful display filters while providing scripting capabilities and integration with external Security Information and Event Management (SIEM) systems. Instead of visualizing the entire packet, TShark can be instructed to extract only specific fields in the form of structured text (CSV/JSON), drastically accelerating analysis. Terminal automation enables investigative teams to quickly profile traffic, extract lists of visited domains, used TLS certificates, or HTTP User-Agent strings, which can subsequently be cross-referenced with Threat Intelligence databases for Indicators of Compromise (IoCs) [6].

5. Practical experiment: Session recomposition and artifact extraction

To demonstrate the practical significance of the developed methodology for passive network analysis and Deep Packet Inspection (DPI), a practical forensic experiment was

conducted. The experiment was carried out in a pre-configured laboratory topology (isolated subnet 192.168.8.0/24), where network traffic between the client node (192.168.8.100) and the gateway (192.168.8.1) was passively captured via a Throwing Star LAN Tap Pro and recorded in the background by a Raspberry Pi 4B into a raw capture.pcap file. The experiment encompasses two main scenarios: the analysis of unencrypted traffic with application artifact extraction, and the analysis of encrypted TLS/SSL sessions. Unencrypted communication protocols (HTTP, DNS, FTP, SMB) provide full visibility into the transmitted data. The investigation process proceeds through the following sequential stages:

• Profiling DNS Queries: The first step in reconstructing the incident timeline is the analysis of DNS resolution. Using the dns filter in Wireshark (or tshark -Y "dns.flags.response == 0") allows the isolation of all outgoing requests from the host.

• Extracted Artifact: A DNS query to the domain test-site.lab was registered, followed by an A-record resolving to IP address 192.168.8.150. This identifies the target of the subsequent communication.

• Extracting HTTP Headers and POST Parameters: By applying the http.request filter, requests to the web server are isolated. Forensic packet dissection yields key metadata:

• User-Agent: Identifies the operating system and browser version (or malicious script/bot) that initiated the connection.

• HTTP POST Content: Using the Follow TCP Stream tool, the raw stream is reassembled into its original form. A submitted HTTP POST request to /login.php containing a form with unencrypted user credentials (username=admin&password=Password123!) was identified.

• File Carving / File Extraction: A critical task of DPI is discovering exfiltrated documents or downloaded malware. In Wireshark, the Export Objects -> HTTP (or SMB/FTP) functionality is utilized.

Result: The software dissector identifies a binary stream transfer with MIME type application/x-msdownload (an .exe executable file). The file was successfully extracted from the .pcap dataset, saved to the analytical station, and its cryptographic hash sum (SHA-256) was computed for cross-referencing with malware databases (VirusTotal):

• sha256sum extracted_binary.exe

• Extracted hash:
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855

The increasing adoption of encrypted transport and application-layer protocols, including DNS over HTTPS (DoH), further limits the visibility of traditional passive packet inspection by concealing both application payloads and, in some cases, DNS resolution traffic [3].

The majority of modern network traffic utilizes encryption (TLS 1.2/1.3), making the payload unreadable without the availability of appropriate cryptographic secrets for session decryption [5]. Nevertheless, applying passive DPI to the unencrypted phase of the TLS handshake yields a significant amount of security intelligence:

• SNI (Server Name Indication) Extraction: During the transmission of the Client Hello packet, the client sends the unencrypted SNI extension, indicating the exact hostname requested. Via the tls.handshake.extensions_server_name filter, the analyst determines the exact FQDN (e.g., malicious-c2-server.com) the client is attempting to access, even if the subsequent HTTP traffic itself is fully encrypted.

• Server Certificate (X.509) Inspection: In the Server Hello message (in TLS 1.2 and the unencrypted portion of TLS 1.3), the server sends its digital certificate.

• Extracted Parameters: Certificate Issuer, validity period, serial number, and Subject Alternative Name (SAN) list. Detecting self-signed certificates or mismatched SAN fields serves as a strong indicator of malicious activity or Man-in-the-Middle (MITM) infrastructure.

• Client Fingerprinting via JA3 / JA4 Hashes: JA3 is a standard for passively creating unique digital fingerprints of clients, based on parameters from the Client Hello message (TLS version, supported cipher suites, supported extensions, and elliptic curves). [8]

• Methodology: The structure of these fields does not depend on the user, but rather on the specific application or library (e.g., Python requests, Go-client, standard Chrome browser, or a specific Trojan).

Result: By generating a JA3 hash sum from the captured packet:

• JA3 String: 771,4865-4866-4867-49195-49199,0-23-65281-10-11,29-23-24,0

• JA3 Hash (MD5): ada702c2db2e7b1a20065c717e3f21e1

6. Conclusion

This report demonstrates that passive network analysis and Deep Packet Inspection (DPI) are indispensable components in the arsenal of modern digital forensics. By implementing the Full Packet Capture (FPC) model and adhering to a strict methodology for maintaining data integrity, investigative teams possess a reliable mechanism for reconstructing the timeline of cyber incidents. The use of specialized software (tcpdump, Wireshark, TShark) enables effective filtering, session recomposition, and the extraction of Indicators of Compromise (IoCs) without altering the state of the investigated environment or alerting potential attackers.

References

[1] Bejtlich, R. (2013). The Practice of Network Security Monitoring: Understanding Incident Detection and Response. San Francisco.

[2] Davidoff, S., & Ham, J. (2012). Network Forensics: Tracking Hackers Through Cyberspace. New Jersey.

[3] Hoffman, P., & McManus, P. (2018). DNS Queries over HTTPS (DoH). RFC 8484, Internet Engineering Task Force (IETF).

[4] Iyengar, J., & Thomson, M. (2021). QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000, Internet Engineering Task Force (IETF).

[5] Rescorla, E. (2018). The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, Internet Engineering Task Force (IETF).

[6] Sanders, C. (2017). Practical Packet Analysis: Using Wireshark to Solve Real-World Network Problems (3rd ed.). San Francisco

[7] Tcpdump Group. (2023). Tcpdump/Libpcap Official Documentation. [Online]. Available: https://www.tcpdump.org/

[8] Althouse, J., Atkinson, J., & Brady, J. (2017). JA3 - A method for profiling SSL/TLS Clients. Salesforce Engineering. [Online]. Available: https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-24736285ea35/

[9] Elitsa Kuzmanova, Donika Dimanova, Technical privacy risks in Generative AI (GenAI) – Lessons from OWASP and Real-World Failures, 10th International Conference on Energy Efficiency and Agricultural Engineering (EE&AE 2025), 5-7

November 2025, Starozagorski Bani, Bulgaria, 979-8-3315-1299-6/2025 IEEE. [Online]. Available: https://doi.org/10.1109/EEAE65901.2025.11273791.
[10] Wireshark Foundation. (2023). PCAP Next Generation (pcapng) Capture File Format. [Online]. Available: https://www.wireshark.org/ / https://tshark.dev/.