Forensic Science International: Digital Investigation 32 (2020) 200892

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

# Packet analysis for network forensics: A comprehensive survey

Leslie F. Sikos

*Edith Cowan University, Australia*

ARTICLE INFO

*Article history:*
Received 16 May 2019
Received in revised form
27 August 2019
Accepted 1 October 2019
Available online 20 February 2020

*Keywords:*
Packet analysis
Deep packet inspection
Network forensics
Packet sniffer
Wireshark
Pcap
Digital evidence
Network monitoring
Intrusion detection

ABSTRACT

Packet analysis is a primary traceback technique in network forensics, which, providing that the packet details captured are sufficiently detailed, can play back even the entire network traffic for a particular point in time. This can be used to find traces of nefarious online behavior, data breaches, unauthorized website access, malware infection, and intrusion attempts, and to reconstruct image files, documents, email attachments, etc. sent over the network. This paper is a comprehensive survey of the utilization of packet analysis, including deep packet inspection, in network forensics, and provides a review of AI-powered packet analysis methods with advanced network traffic classification and pattern identification capabilities. Considering that not all network information can be used in court, the types of digital evidence that might be admissible are detailed. The properties of both hardware appliances and packet analyzer software are reviewed from the perspective of their potential use in network forensics.

© 2019 The Author. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

## 1. Introduction to packet analysis in network forensics

The ever-increasing popularity of online services urges security experts and law enforcement agencies to find new ways of investigating cybercrimes and finding evidence admissible in court. Online services transfer large amounts of data over communication networks in a variety of forms, among which *network packets* are the most common. These are groups of bits that include data complemented by control information (Stallings and Case, 2012), generally referring to a network layer (OSI Layer 3) protocol data unit. They represent the smallest unit of data at a particular point in time intercepted and logged about network traffic flow traversing over packet-switched networks,[1] consisting of control information (source and destination IP address, error detection codes, sequencing information) and *payload* (intended message). A data unit in OSI Layer 2 (data link layer) is called a *frame*, which is a group of bits that includes data with one or more addresses and other protocol control information (Stallings and Case, 2012). In OSI Layer 4 (transport layer), the equivalent is called a *segment* (or *datagram*).

Network packets, when captured, stored, and processed efficiently, can be used in forensic investigations and may even provide admissible evidence against a suspect in a court case. Note that throughout this paper, we are going to use the term packet analysis, irrespective whether the actual content is a frame, packet, datagram, or session, unless stated otherwise.

## 2. Capturing and storing network packets

The communication between network devices are facilitated using *protocols*, i.e., mechanisms to identify and establish connections, and formatting rules and conventions specified for data transfer. Network data can be analyzed, and network traffic segregated by type, using purpose-built software. Those *protocol analyzers* that are designed for packet analysis are called *packet analyzers* (*packet sniffers*, sometimes *network analyzers*). These software tools intercept and log network traffic traversing over a digital network or a part of a network through the process of *packet capturing*. The captured packets can then be analyzed by decoding the raw data of the packets and visualized via displaying various fields to interpret the content (Chapman, 2016).

By putting a capable wired network interface controller (NIC) or

*E-mail address:* l.sikos@ecu.edu.au.

[1] A *packet-switched network (PSN)* is a type of communication network that groups and sends data in the form of small packets, thereby enabling data/network packet sending between a source and a destination node over a shared data path (network channel) between multiple users and/or applications.

https://doi.org/10.1016/j.fsidi.2019.200892
1742-2876/© 2019 The Author. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

wireless network interface controller (WNIC) into *promiscuous mode*, all received network traffic can be passed to the central processing unit (CPU) rather than just those frames the controller is specifically programmed to receive. Available on most Linux distributions, the *Berkeley Packet Filter (BPF)* supports packet filtering, such as receiving only those packets that initiate a TCP connection. Because BPF only returns the packets that pass the filter, irrelevant packets do not have to be copied from the operating system to the kernel to be processed, thereby greatly improving performance. An enhancement of the original BPF is *Extended BPF (eBPF)*, which supports not only forward jumps, but also backward jumps, thereby allowing loops. Using global data stores called *maps*, eBPF can also be used for aggregating event statistics.

There are various approaches for "tapping into the wire"; which one to use depends on the networking environment in which the device whose traffic should be analyzed is located. In those—rather rare—networks that use hubs, a packet sniffer can see all the devices in the network, simply because traffic sent through a hub goes through every port connected to that hub. In a switched networking environment, the visibility from a packet sniffer is limited to the port we plug into. On switched networks, there are four main ways to capture traffic from a target device: port mirroring (port spanning), hubbing out, using a tap, and ARP cache poisoning (ARP spoofing). Which one to choose depends on the case: the first one is an option only if we have access to the command-line or web-based management interface of the switch on which the target computer is located, the switch supports port mirroring and have an empty port into which we can plug our sniffer; the second one needs physical access to the switch the target device is plugged into; the third one requires a special hardware tool (network tap) to be connected to the network; and the fourth one requires information to be collected, such as the IP address of the analyzer system, the remote system from which we would like to capture the traffic, and the router from which the remote system is downstream.

Network packets hold useful information about network activities, and analyzing them helps in gathering and reporting network statistics and debug client-server communications. Network packet capture files store a lot of information about online user activity that can be useful in network forensics, such as visited websites and the time spent on browsing them,[2] successful and unsuccessful login attempts, credentials, illegal file downloads, intellectual property abuse, etc. Packet files not only contain a wealth of information, but data can be retrieved from them in various groupings, such as individual frames, client-server conversations, packet streams, flows, and sessions. In network forensics, packet analysis can be used to collect evidence for investigations of digital activities, and to detect malicious network traffic and behavior, including intrusion attempts and network misuse, and identify man-in-the-middle attacks and malware such as ransomware (Alhawi et al., 2018).

The de facto standard capture format is *libpcap (pcap)*, which is a binary format that supports nanosecond-precision timestamps.[3] Although this format varies somewhat from implementation to implementation, all pcap files have the general structure shown in Fig. 1.

The global header contains the *magic number* (to identify the file format version and byte order), the GMT offset, the timestamp precision, the maximum length of captured packets (in octets), and the data link type. This information is followed by zero or more records of captured packet data. Each captured packet starts with the timestamp in seconds, the timestamp in microseconds, the number of octets of packet saved in file, and the actual length of the packet.

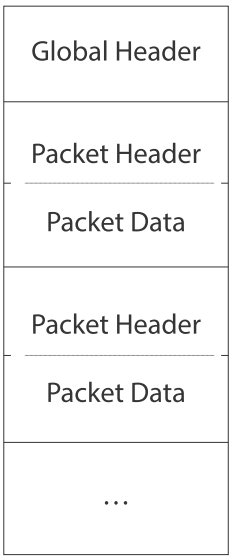


**Fig. 1.** The general structure of pcap files. The packet data contains at most the first *N* bytes of each packet, where *N* is the snapshot length (typically smaller than 65,535).

The successor of pcap is the *pcap Next Generation Capture File Format (pcapng)*. Rather than being limited to dumping network packets only, pcapng allows for saving a range of data types using a generic block format. The structure of pcapng files, shown in Fig. 2, is developed by the IETF.

The details of the block structure depends on the block type; the list of block types includes section header blocks, interface description blocks, simple and enhanced packet blocks, name resolution blocks, interface statistics blocks, systemd journal export blocks, decryption secrets blocks, and custom blocks. Further types are under development.

The *snoop* capture format is defined in IETF RFC 1761.[5] Each snoop file is an array of octets consisting of a fixed-length file header and one or more variable-length packet records. Each file header contains a 64-bit identification pattern, a 32-bit version number, and a 32-bit datalink type.

The *RedBack Smartedge* pcap format (SE400/800) was designed for PowerPC-based NetBSD and intelligent packet-forwarding linecards. This format is based on circuits and extends the pcap format with additional informations about protocols and circuits.

Further capture formats include the following: InfoVista 5View capture, the *IxCatapult* (formerly *DCT 2000*) trace.out file format, the Cisco Secure IDS iplog format, the Symbian OS btsnoop format, the TamoSoft CommView format, the Endace ERF capture format, the EyeSDN USB S0/E1 ISDN trace format, HP-UX nettl trace, the K12 text file format, the Microsoft Network Monitor (NetMon) format, the NA Sniffer format, the Network General Sniffer format, the Network Instruments Observer format, the NetXray format, the Novell LANalyzer format, the *PDML* Packet Description Markup

[2] In a corporate environment, these can also be retrieved from other sources, such as firewall logs.

[3] This precision is often not available in practice, because the packet capture implementation in place may support only milliseconds, such as if the timestamp is added by the kernel or the CPU the capture is offloaded to, or if a packet has been waiting in a ring buffer.

[4] https://xml2rfc.tools.ietf.org/cgi-bin/xml2rfc.cgi?url=https://raw.githubusercontent.com/pcapng/pcapng/master/draft-tuexen-opsawg-pcapng.xml

[5] https://tools.ietf.org/rfcmarkup?rfc=1761

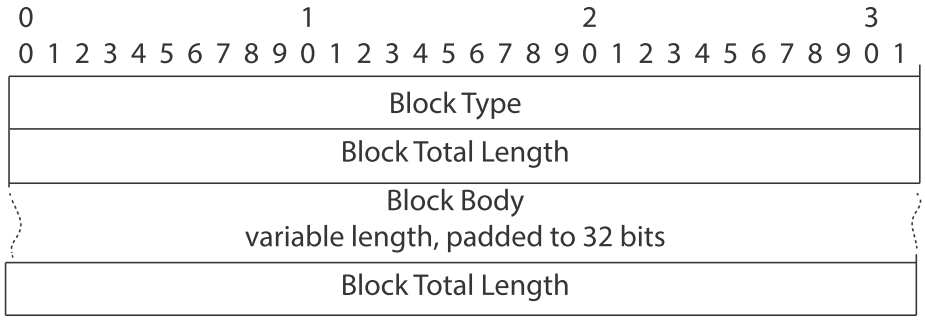


**Fig. 2.** The basic block structure of pcapng files.[4]

Language file format, the NetScaler Trace format, the RADCOM WAN/LAN Analyzer format, the Shomiti/Finisar Surveyor format, the Sniffer Pro format, the Tektronix K12xx.rf5 format, and the Visual Networks UpTime traffic capture format.

## 3. Processing network packets and packet flow

Network packets hold more than just communication data and metadata; files that traversed through a network can be reconstructed from network packet streams (*network carving*) (Beverly et al., 2011) using purpose-designed network carvers or packet analyzers that support file export from packet capture. This, together with other options to find traces of network data transfer, makes *packet analysis* a primary traceback technique in network forensics. It can assist in finding traces of nefarious online behavior and breaches affecting an organization, determining the source of network security attacks, and acquiring host-based evidence of malicious actions (Johansen, 2017), although making sense of encrypted network traffic is far more challenging than the analysis of unencrypted traffic (van de Wiel et al., 2018). For example, network traffic classification based on packet analysis and port numbers alone is infeasible for encrypted VoIP applications, such as Skype (Alshammari and Zincir-Heywood, 2015), although even encrypted network traffic can be classified using machine learning (Dong and Jain, 2019).

*Packet sniffing* is a method of tapping packet flows, i.e., packets as they flow across a communication network (Ansari et al., 2003), and even re-transmitted packets, such as with different TCP properties. This can be utilized for reconstructing data transferred over the network, and might even be used as an anti-forensic measure.

### 3.1. Deep packet inspection

*Deep packet inspection (DPI)* refers to a type of packet analysis that goes beyond packet header information and analyzes the packet payload as well. DPI can be used to identify excessive levels of non-business traffic in enterprises such as social media use that need to be filtered or blocked, to detect data streams (Yin et al., 2018), video traffic (Huang et al., 2012), encrypted BitTorrent traffic (Carvalho et al., 2009), malicious behavior (Guo et al., 2017), malicious traffic (Stergiopoulos et al., 2018), intrusions (Parvat and Chandra, 2015), etc., to detect hosts behind Carrier-Grade NATs by extracting non-routable IP addresses from peer lists obtained by crawling the BitTorrent Distributed Hash Table (DHT) (Richter et al., 2016), to classify malware (Boukhtouta et al., 2016), to analyze honeypot traffic (Pimenta Rodrigues et al., 2017), to facilitate forensic-driven security monitoring, and to enable forensic-by-design industrial systems (Parra et al., 2019). In fact, deep packet inspection can reveal and record online activities to the extent that it raises privacy concerns regarding mass surveillance by state and government agencies (particularly under legislations that require "wiretap-friendly" online services, such as CALEA in the U.S.),[6] even if the sheer volume of traffic makes it impractical to record all traces of user activity (Stalla-Bourdillon et al., 2014). On the bright side, deep packet inspection allows network operators to shape traffic and control various types of traffic, such as email, VoIP, and P2P. Companies such as NETSCOUT[7] and Sandvine[8] provide DPI services to prioritize network activity, enforce policies, and to help develop new service plans.

### 3.2. Using artificial intelligence in packet analysis

Formal knowledge representation is applied in network forensics in the form of ontologies to automate the processing of network packet sequences (Sikos, 2018). Purpose-built ontologies, such as the *Packet-Centric Network Ontology (PACO)* (Ben-Asher et al., 2015) and the *Packet Analysis Ontology (PAO)*[9] (Sikos, 2019), can capture the semantics of actual network packets, and provide terms to formally describe background knowledge in a machine-interpretable form. The datasets that utilize these definitions and codified expert knowledge, together with reasoning rules, can be used to infer new statements and make implicit network knowledge explicit.

To decrease the rate of false positives when detecting malicious traffic using Snort, Shah and Issac (2018) applied machine learning, and implemented a plugin. This plugin decodes packet data, classifies network packets, uses SVM, decision tree, a combination of SVM and fuzzy logic, and optimized SVM with firefly, to differentiate between legitimate and malicious traffic, and reduce the rate of false positive alarms.

Deep packet inspection combined with semi-supervised machine learning is suitable for efficiently classifying flows to identify audio, video, and interactive data, thereby facilitating fine-grained adaptive QoS traffic engineering (Yu et al., 2018). A periodic retraining using a dynamic flow database enables the classifier to adapt to rapidly and constantly changing network traffic characteristics.

The deep learning-based approach of Lotfollahi et al. (2019), called *Deep Packet*, integrates feature extraction and classification. It can characterize network traffic into classes such as FTP and P2P using two deep neural network structures, *stacked autoencoder (SAE)* and *convolutional neural network (CNN)*, and can identify end-user applications such as Skype and BitTorrent. The Deep Packet approach can not only identify encrypted traffic, but can also

[6] Communications Assistance for Law Enforcement Act.
[7] https://www.netscout.com
[8] https://www.sandvine.com
[9] https://purl.org/ontology/pao/

distinguish between VPN and non-VPN network traffic.

#### 3.2.1. Optimizing and offloading packet processing

Hardware acceleration and offloading for network packet processing is available via application-specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), and graphics processing units (GPUs). For example, FortiGate's FortiASIC NP6 supports offloading of most IPv4 and IPv6 traffic, IPsec VPN encryption, CAPWAP traffic, and multicast traffic.

### 3.3. Programming packet processing applications

Performing network packet analysis, and deep packet inspection in particular, with speeds in the Gbps range requires specialized hardware, which is typically programmed in Assembly or C (Duncan and Jungck, 2009). An alternative approach is to use the purpose-designed *packetC* programming language with a parallel packet processing model. This language provides high-level constructs to express coding solutions for packet processing applications. Compared to C, it simplifies and constrains type declarations to prevent unforeseen type conflicts, avoids type coercions or promotions to prevent unexpected data truncations or expansions, and supports a strong typing model with restrictive type casting to prevent unexpected side effects (Jungck et al., 2011).

## 4. Packet data as digital evidence

The capture, analysis, and backtracing of network packets constitute a considerable part of network forensics (Nikkel, 2005). Network packets are sources of network evidence, and together with data from remote network services, form *live network evidence sources*. Depending on the online content, network packets have a finite, non-zero acquisition window during which evidential data can be observed or acquired. On the one hand, some argue that using packets as evidence can be problematic in case they are spoofed (Kim et al., 2015). On the other hand, network packets can complement firewall logs and network monitoring software extremely well, and can be considered the ultimate forensic evidence (Hurd, 2018).

Packet capture files can be used to extract potential forensic evidence from network data, such as via the *Highly Extensible Network Packet Analysis (HENPA)* framework (Broadway et al., 2008). The information extracted from network packets can be used as evidence either directly or indirectly. For example, some information contained in the packets, including the sender and receiver IP addresses, port numbers, etc., along with the transferred data, can be used directly as evidence. Inferred, indirect information derived from multiple packets that can be used as evidence include patterns such as large streams of ICMP packets send from a particular host to another one in a short period of time, which might indicate a denial-of-service (DoS) attack.

The utilization of packet analysis to its full potential relies on *full packet capture*,[10] which requires a full telecommunication interception warrant or equivalent (Turnbull and Slay, 2007). Examples of corresponding legislations that detail the requirements for these include ETSI's *Council Resolution of 17 January 1995 on the Lawful Interception of Telecommunications*,[11] the Australian *Telecommunications (Interception and Access) Act*[12] and the *Surveillance Devices Act*.[13] If the necessary warrant is obtained and full packet capture is performed (which is often not feasible due to network bandwith constraints), security engineers can play back all the traffic on a network (Rounsavall, 2017).

Because packet capture files often contain sensitive data, such as personal data of network users, information about the internal structure of an enterprise network, etc., privacy restrictions, policies, and laws make it impossible to share packet capture files. There are approaches and solutions to automatically scramble network packet capture data while preserving binary integrity, such as *SafePcap*,[14] which complies with the Europen Union's *General Data Protection Regulation (GDPR)*[15] and NIST's *NISTIR 8053 "De-Identification of Personal Information."*[16] SafePcap performs data modifications in a break-proof manner by recalculating the lengths, checksums, offsets and all other services for all affected packets and protocol layer fields on the fly.

A full packet capture is imperative when investigating what has happened in a network at a particular point in time and who was actually involved in an online activity, because the IP address of a suspect's computer alone cannot serve as the basis of forensic investigations due to the dynamic nature of IP addresses, and because they often cannot be linked directly to an individual (Clarke et al., 2017) and often not even to an exact geographical location (Afanasyev et al., 2011). Nevertheless, following the TCP stream of the simultaneous use of SMTP and a particular IP address can identify the address associated with the From tag of the email header. Furthermore, email headers contain the name of the sender, which may reveal the suspect's real name. Emails sent by the user can be reconstructed, including any attachments. The manufacturer of a suspect's computer can be identified with a high certainty based on the Organizational Unique Identifier (OUI) part of the device's MAC address,[17] although this cannot be used in many cases, particularly in corporate networks. Based on the packet data, it can be determined when the suspect logged in to the network. If the password of the suspect was encoded in Base64, it can be converted to UTF-8 to reveal the actual password that was used to log in. Ultimately, such information can help build a profile of the suspect's identity.

Supporting evidence can be efficiently collected from stored packet information by recreating the original metadata, files, or messages sent or received by a user (Manesh et al., 2011). The analysis of file and software downloads can help identify *drive-by downloads* leading to malware infections (Ndatinya et al., 2015). Malicious online activities may be identified based on common traits of SQL queries used for attacks on TCP, such as SYN flood attacks, XMAS scans, and SYN/FIN attacks (Kaushik et al., 2010). What level of forensic evidence can be obtained depends on the tradeoff set between packet file details and network throughput (Ning et al., 2013).

Attackers might try to hide attacks by forging packets (Kim and Kim, 2011). Nevertheless, in many cases, the origin of packets can be traced back with techniques such as *link testing* (e.g., *controlled flooding* (Burch and Cheswick, 2000) or *input debugging* (Stone, 2000)), *ICMP traceback* (Bellovin and Leech, 2000), *hash-based IP traceback* (Snoeren et al., 2001), *single-packet IP traceback* (Snoeren et al., 2002; Murugesan et al., 2018), *coordinated packet traceback* (Sy and Bao, 2006), *packet marking* (such as by reconstructing the attack path (node-append, node sampling, or edge sampling

[10] Flow records, particularly when machine learning is employed, can be used for traffic classification comparably well (Dong and Jain, 2019).

[11] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.C_.1996.329.01.0001.01.ENG&toc=OJ:C:1996:329:TOC

[12] https://www.legislation.gov.au/Series/C2004A02124

[13] https://www.legislation.gov.au/Series/C2004A01387

[14] https://omnipacket.com/safepcap

[15] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679

[16] https://csrc.nist.gov/publications/detail/nistir/8053/final

[17] The first half (first 24 bits) of the MAC address.

(Savage et al., 2001)), using an ID storing scheme (*k-fragmentation* (Savage et al., 2001), *k-segmentation* (Xiang et al., 2008), or *hash digest* (Song and Perrig, 2001))), or via the combination of packet marking and logging (Gong and Sarac, 2005; Al-Duwairi and Govindarasu, 2006).

## 5. Network packet analyzers

Generally, each packet analyzer performs four steps to process packets (Yang et al., 2018):

1. *Open a packet capture socket:* select a network device and open it for live capture, retrieve the network address and subnet mask, convert the packet filter expression into a packet filter binary, and assign the packet filter to the socket
2. *Packet capture loop:* determine the datalink type and start the packet capture
3. *Parse and display packets:* set a character pointer to the beginning of the packet buffer and move it to a particular protocol header by the size of the header preceding it in the packet, and map the header to the appropriate header structure (IP, TCP, UDP, ICMP, etc.) by casting the character pointer to a protocol-specific structure pointer
4. *Terminate the capturing process:* send interrupt signals and close the packet capture socket

Packet analyzers are designed for various purposes and differ in terms of capabilities and features, hardware resource utilization, processing speed (Goyal and Goyal, 2017), supported protocols, user-friendliness, supported operating systems, supported network types, interface, license, and implementation type. Many packet analyzers support both live capture and offline analysis. The deep inspection of packets and the analysis of various types of network traffic are available only in those analyzers that support hundreds of protocols. Those packet analyzers that intercept traffic on wireless networks are called *wireless analyzers* (*WiFi analyzers*), e.g., Aircrack-ng,[18] and Kismet.[19] For Bluetooth, there is a purpose-built packet sniffer called *FTS4BT*.[20]

Some tools support data carving, capture file quality assessment, anomaly detection, protocol encapsulation, and flexible packet aggregation. The list of supported file formats varies between packet analyzers, and some tools even provide on-the-fly gzip decompression.[21]

The analyzers that come with a GUI feature typically have a packet browser to visualize the packet content, and various display filters to show only the information relevant for a particular task, rather than everything captured. Some packet analyzers can differentiate between frame types, and visualize them using color schemes.

In terms of licensing, packet analyzers are either open source, freeware, or commercial. Common license types associated with packet analyzers include the GNU General Public License[22] and proprietary licenses.

There are both hardware appliances and software implementations for packet analysis, although software tools are far more common than hardware implementations.

### 5.1. Hardware devices for packet analysis

Notable hardware packet analyzers include the *Fluke Lanmeter* series (now discontinued), *PNtMS* (Rahman et al., 2009), the packet analyzer of Thomas et al. (2011), *KPAT* (Wang et al., 2014), the embedded packet logger of Jandaeng (2016), the *Cisco Security Packet Analyzer* appliances,[23] SolarFlare's SolarCapture appliances,[24] Corvil's hardware appliances,[25] IPCopper's packet capture appliances,[26] *RSA Netwitness*,[27] *Arbor Ellacoya e100*,[28] Sandvine's *PacketLogic* platforms,[29] the *ExtraHop Discover Appliances*,[30] and LiveAction's *LiveCapture* (formally *Omnipliance Ultra*) devices.[31]

Some of these appliances are physical (embedded or dedicated), others are bare-metal, virtualized, or hybrid.

### 5.2. Packet analyzer software

Among the packet analyzer software tools, there are purpose-designed packet analyzers and network tools that provide features for packet capture and analysis. Such network tools include intrusion detection software, proxies, vulnerability assessment tools, network scanners, and network monitoring tools, which are used in network forensics (Joshi and Pilli, 2016).

In 1997, the Federal Bureau of Investigation (FBI) implemented its customizable packet sniffer as part of the system called *Carnivore* (which was later renamed to *DCS1000*). It monitored users' Internet traffic, including emails. It was phased out by 2005. In 1998, Gerald Combs developed *Ethereal*, a free and open-source packet analyzer, which was renamed to *Wireshark* in 2006 (Orebaugh et al., 2006). Over the years, Wireshark has become one of the most widely used graphical packet capture and protocol analysis tools (Shimonski, 2013), featuring a highly intuitive GUI for packet analysis (Sanders, 2017). This GUI has a customizable packet browser that displays a maximum of three panes simultaneously, including a packet list and the packet details and packet bytes of the currently selected packet (see Fig. 3).

The program features coloring rules to differentiate between inactive and active selected items, marked packets, ignored packets, follow streams (client and server), and to display valid, invalid, and warning filters. It also has efficient display filters to focus on those frames that are relevant for a particular analysis, such as by showing HTTPS traffic only or network communication related to a particular IP address. The functions of Wireshark are also available in a command-line tool called *TShark*, and Wireshark provides additional tools for managing packet captures (*capinfos*, *mergecap*, *editcap*). Due to its versatile functionality, Wireshark is widely deployed and is in the focus of attention of practitioners and researchers alike (Mielczarek and Mon, 2015; Das and Tuna, 2017; Alsmadi et al., 2018; Islam et al., 2018; Bhandari et al., 2017).

Also in 1998, Martin Roesch introduced *Snort*,[32] a free open source network intrusion detection/prevention system (IDS/IPS), which can capture and analyze network traffic. It is now developed by Cisco Systems, and can perform real-time traffic analysis and

[18] https://www.aircrack-ng.org
[19] https://www.kismetwireless.net
[20] http://www.fte.com/products/FTS4BT.aspx
[21] It is a common practice to compress large capture files using gzip.
[22] https://www.gnu.org/licenses/gpl.html
[23] https://www.cisco.com/c/en/us/products/collateral/security/security-packet-analyzer/datasheet-c78-737589.pdf
[24] https://solarflare.com/solarcapture/
[25] https://www.corvil.com/products/technology/appliances
[26] http://www.ipcopper.com/forensic_packet_capture.htm
[27] https://community.rsa.com/community/products/netwitness/hardware-setup-guides
[28] http://www.lextel.com/documents/E100Datasheet.pdf
[29] https://www.sandvine.com/products/packetlogic
[30] https://www.extrahop.com/products/appliances/
[31] https://www.liveaction.com/products/live-capture/
[32] https://www.snort.org

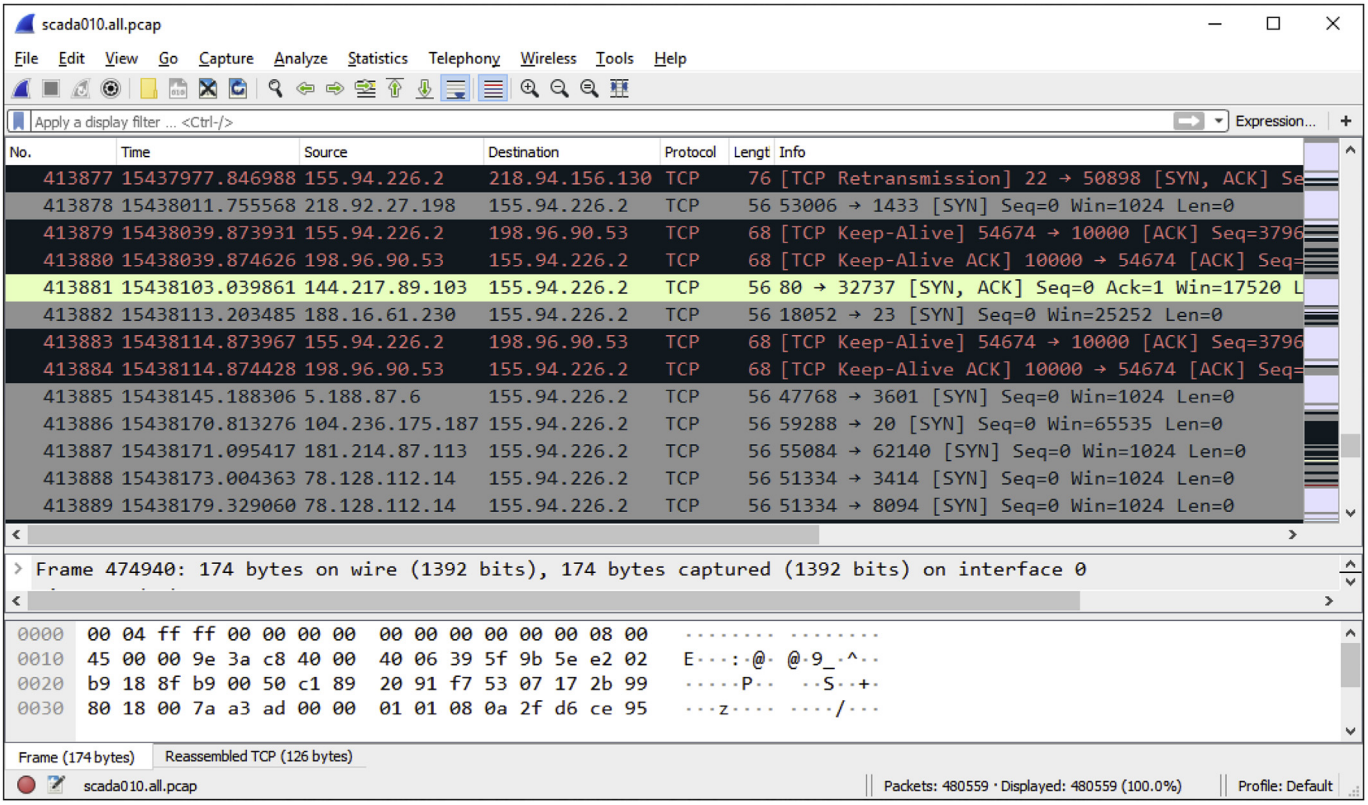


**Fig. 3.** Wireshark can colorize packets by type and displays them in context.

packet logging on IP networks. Snort utilizes detection rules for various kinds of network traffic, and is proven to be useful in complex network behavior analysis, such as to help detect advanced persistent threats (APTs) (Cui et al., 2018). Similar to Wireshark, Snort is actively developed, and many third-party plugins are available that extend its features, such as *VisSRA*, which visualizes Snort rules and alerts (Hong et al., 2012).

In 1999, Eddie Kohler (at MIT at the time) developed *ipsumdump*[33] to summarize TCP/IP dump files or other packet source into a self-describing ASCII format for human and machine consumption.

*Dsniff*[34] is part of a collection of tools for network auditing and penetration testing, developed by Song for Unix-like operating systems. It parses various application protocols and extracted relevant information, for example, it decodes passwords sent in cleartext across a switched or unswitched Ethernet network.

In 2000, Toledo developed *EtherApe*,[35] a packet sniffer and network traffic monitoring tool, for Unix. This free and open source software visualizes network traffic using graphs, in which each node represents a specific host, and the edges of the graphs are the network connections. The color-coding of nodes and links makes it possible to differentiate between various protocols. The amount of network traffic is represented proportionally with the width of the graph edges, as shown in Fig. 4.

*Tcpdump*[37] is a command-line tool that has been around for about two decades and is one of the de facto standard tools for capturing and dumping network packets for later analysis. It also has a Windows implementation called *WinDump*.[38] Tcpdump is developed hand in hand with *libpcap*,[39] a popular packet capture software library to capture live network traffic data, which is also utilized by both packet analyzers and other software with packet sniffing capabilities, such as Wireshark and Snort. In turn, libpcap employs the *pcap* API. The pcap API is also used by the *WinPcap*[40] and *Npcap*[41] packet sniffers, and *ngrep*,[42] a tool that can match regular expressions within the network packet payloads.

In 2001, ALoR and NaGA developed *Ettercap*,[43] a free and open source network security tool for man-in-the-middle (MITM) attacks on LANs. It displays the IP and MAC address of all hosts connected to a network. It can identify hosts having unauthorized IP addresses and thereby detecting attackers, although if an attacker uses an authorized IP address via IP spoofing, this will not be detected (Agrawal and Tapaswi, 2017).

The *Charles Web Debugging Proxy*,[44] developed by Karl von Randow in 2002, is an HTTP proxy, HTTP monitor, and reverse proxy that visualizes all the HTTP and SSL/HTTPS traffic between a computer and the Internet. In the following year, Eric Lawrence developed *Fiddler*,[45] a free web debugging proxy that can log HTTP/HTTPS traffic. The captured network data can be filtered to hide sessions, highlight the traffic of interest, bookmark breakpoints, etc.

[33] http://www.read.seas.harvard.edu/~kohler/ipsumdump/
[34] https://www.monkey.org/~dugsong/dsniff/
[35] https://etherape.sourceforge.io
[36] https://etherape.sourceforge.io/images/v0.9.3.png
[37] https://www.tcpdump.org
[38] https://www.winpcap.org/windump/
[39] https://www.tcpdump.org
[40] https://www.winpcap.org
[41] https://nmap.org/npcap/
[42] https://github.com/jpr5/ngrep/
[43] http://www.ettercap-project.org
[44] https://www.charlesproxy.com
[45] https://www.telerik.com/fiddler

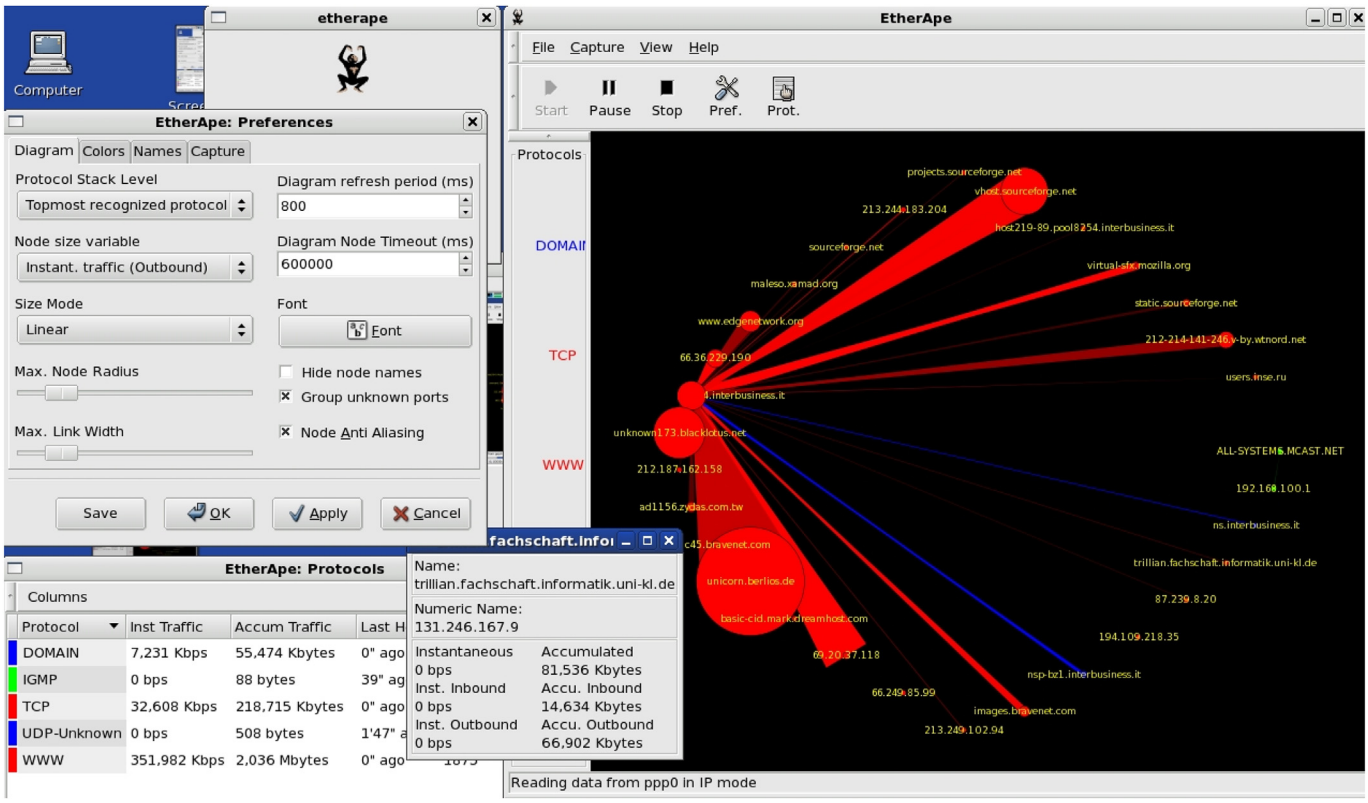


**Fig. 4.** The EtherApe GUI represents network connections as graphs.[36]

The software features a Session Inspector widget that can display the contents of a recorded web session, including status, headers, caching, cookies, URLs, protocols, type of compression used, redirects, and so on.

In 2009, the Open Information Security Foundation released *Suricata*,[46] an open source-based intrusion detection and intrusion prevention system (IDS/IPS), which can, among other features, scan pcap files with IDS rulesets to find traces of suspicious or malicious network activities. Similar to Snort, Suricata is popular enough to have support for many third-party tools that can complement it for visualization and analysis, including *Snorby*,[47] *BASE*, *Sguil*,[48] *u2platform* (formerly *Aanval*),[49] and *CERNE*.[50]

Also in 2009, Daniel Borkmann wrote *netsniff-ng*,[51] a free Linux network analyzer. It is a high-performance tool that utilizes zero-copy mechanisms for network packets, making it unnecessary for the Linux kernel to copy packets from kernel space to user space via system calls. The tool supports standard pcap files and can also capture, analyze, and replay raw 802.11 frames.

*WebScarab*[52] is an easy-to-use integrated penetration testing tool for finding vulnerabilities in web applications. It has packet analyzing capabilities, although these are limited to applications communicating using the HTTP and HTTPS protocols.

*Tranalyzer*[53] is a free software for flow- and packet-based traffic analysis and network troubleshooting. It is built atop the libpcap library, and it accepts not only IPv4 and IPv6, but also Layer 2 and encapsulated packets, such as MPLS, L2TP, and GRE, from standard pcap files or live interfaces (Burschka and Dupasquier, 2016).

*Tcptrace*[54] is a tool for analyzing the logs produced by tcpdump, snoop, EtherPeek, HP Net Metrix, and WinDump. *Yet Another Flowmeter (YAF)*[55] represents network information flow, the metadata of which can be used as the input for *yafMeta2Pcap*[56] to create pcap files for a particular flow.

The *SolarWinds Network Performance Monitor*[57] comes with a packet analyzer[58] and also features deep packet inspection,[59] which allows the categorization of network traffic into types based on destination server IP addresses, ports used, and measurement of the total and relative volumes of traffic for each type.

Paessler's *PRTG Network Monitor*[60] includes a vast array of network monitoring features, including *Packet Sniffer Sensor*,[61] a packet capture tool. The packet analyzing capabilities of this tool are limited to the analysis of data packet headers, but it has other features, e.g., showing the bandwidth use of different types of network traffic.

To address the inefficient processing of large packet capture files with traditional packet analyzers running on a single host with limited computing and storage resources, Lee et al. (2011)

[46] https://suricata-ids.org
[47] https://github.com/Snorby/snorby
[48] http://sguil.sourceforge.net
[49] https://adaptive.codes/pages/aanval
[50] https://www.telesoft-technologies.com/cyber/monitoring-visibility-for-incident-response/cerne
[51] http://netsniff-ng.org
[52] https://github.com/OWASP/OWASP-WebScarab
[53] https://tranalyzer.com
[54] https://github.com/blitz/tcptrace
[55] https://tools.netsa.cert.org/yaf/
[56] https://tools.netsa.cert.org/yaf/yafMeta2Pcap.html
[57] https://www.solarwinds.com/network-performance-monitor
[58] https://www.solarwinds.com/topics/packet-analyzer
[59] https://www.solarwinds.com/topics/deep-packet-inspection
[60] https://www.paessler.com/prtg
[61] https://www.paessler.com/manuals/prtg/packet_sniffer_header_sensor

introduced an Apache Hadoop-based packet processing tool, which can open even petabyte-sized packet capture files, thanks to the MapReduce parallel processing paradigm. This tool relies on four novel, representative computation modules for total traffic statistics, periodic flow statistics, periodic simple statistics, and top N statistics.

The packet analyzer of Lee et al. (2012) is dedicated to the fast later access (FLA) and deep packet inspection of network packets using IEC 61850 communication protocols. Considering the large number of packets to process in IEC 61850 networks, this analyzer was designed for communications between the server and client of substation automation systems (SASes), each of which consists of a station, a bay, and a process level. The authors claim that their analyzer significantly outperforms other, general packet analyzers when analyzing such networks.

Another tool for the fast packet extraction for large network traces is *PcapWT*, which adopts the wavelet tree data structure to enable fast search and high compression ratio. PcapWT supports multi-threading, which improves random file read and write performance over Solid State Drives (SSDs) (Kim et al., 2015).

CAIDA's *CoralReef*[62] software suite can collect and analyze data from passive Internet traffic monitors, in real time, or from trace files. A distinctive feature of CoralReef is that it can categorize packet traffic by source autonomous systems (ASes).

*Xtractr*[63] is a hybrid cloud application for indexing, searching, reporting, extracting, and collaborating on pcaps.

*Cisco NetFlow*[64] aggregates statistical information on packets flowing through routing devices. It identifies packet flows for both ingress and egress IP packets.[65]

*Capsa*[66] is a comprehensive packet analyzer with support for more than 300 protocols. It can display in-depth packet decoding information in Hex, ASCII, and EBCDIC. It can reconstruct packet streams, and allow packet capture and analysis to be run at a predefined time or recurrence automatically. In addition, Capsa has built-in tools to create and replay packets.

*Meterpreter* of *Metasploit*,[67] Rapid7's vulnerability assessment tool, can store packets in a ring buffer and export them in standard pcap format for later analysis. It builds upon the *MicroOLAP Packet Sniffer SDK*.[68]

*SmartSniff*[69] is a packet capture tool for TCP/IP packets, which displays captured data as a sequence of client-server conversations in ASCII mode (for text-based protocols, such as HTTP, SMTP, POP3, and FTP) or as Hex dump (for non-text-based protocols). SmartSniff provides three methods for packet capture: raw sockets, using the WinPcap capture driver, and on older systems, using the Microsoft Network Monitor driver.

*Omnipeek*[70] provides packet-based analytics by flows (conversation pairs) and visualizes them in intuitive graphical displays. It supports deep packet analysis and can decode over a thousand protocols.

*Moloch*[71] is a standalone open source, indexed full packet capture software. It stores and exports packets in the standard pcap format. Moloch features a powerful web-based GUI, which can display sessions and session profile information in a tabular format, and the top unique values of fields and network connections as graphs.

*PcapDB*[72] is a distributed, search-optimized full packet capture system. It reorganizes captured packets during capture by flow, indexes packets by flow, and allows flow-based searches. The indices for the captured packets typically occupy less than 1% of the size of the captured data.

*Stenographer*[73] is a full packet capture utility for writing packets to disk with high speeds. It provides specific methods to retrieve only those specific packets that are required for a particular analysis, thereby selectively reading only less than 1% of packet data stored on disk.

*Packet Capture*[74] is an Android app with SSL decryption capability. It can display packet contents in Hex or ASCII.

Networking teams who want to share network packets for *collaborative packet analysis* can use tools such as *CloudShark*[75] or *pcapr*.[76]

### 5.3. Packet builders

Some packet analyzers provide features not only for packet capture and analysis, but also for packet manipulation. However, for modifying packets, there are purpose-designed *packet builders* (*packet crafters*) as well.

*Colasoft Packet Builder*[77] can be used to create custom packets. It provides templates for Ethernet, ARP, IP, TCP, and UDP packets, and allows the user to change the parameters in a decoder editor, hexadecimal editor, or ASCII editor to create packets.

*Hping*[78] allows to send custom TCP/IP packets to network hosts while setting the limit for the number of packets after which the sending or receiving should stop, determining the interval between sending packets, and incrementing or decrementing the TTL for outgoing packets.

*Scapy*[79] is a packet manipulation tool written in Python that enables sending, sniffing, dissecting, and forging network packets, and used in software that probe, scan, or attack networks.

*Network Dump Data Displayer and Editor (Netdude)*[80] is a framework for the inspection, analysis, and manipulation of pcap/tcpdump trace files. It can be used to inspect and filter packets at arbitrary locations in trace files, inspect and edit the values of fields in a protocol's packet header, resize individual packets, directly modify packet payload, define arbitrary trace areas for subsequent packet modifications, and copy and move packets between, and delete packets in, trace files.

*Fragroute* is a command-line packet sniffer that can not only capture packets, but also intercept, modify, and rewrite network traffic, such as by reordering packets or injecting meaningful packets of arbitrary size and length into data streams of TCP/IP sessions (Yang et al., 2018). This can be particularly useful for stepping-stone intrusion detection when analyzing how intruders can manipulate sessions to stay undetected for long periods of time.

[62] http://www.caida.org/tools/measurement/coralreef/
[63] http://www.pcapr.net/xtractr
[64] https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html?dtid=osscdc000283
[65] *Ingress filtering* means checking the source IP field of IP packets a router receives. Packets that do not have an IP address in the IP address block to which the interface is connected are dropped. *Egress filtering* is monitoring and potentially restricting the flow of information outbound from one network to another.
[66] https://www.colasoft.com/capsa/
[67] https://www.metasploit.com
[68] http://www.microolap.com/news/detail.php?ID=1406
[69] https://www.nirsoft.net/utils/smsniff.html
[70] https://www.liveaction.com/products/omnipeek/
[71] https://molo.ch
[72] https://github.com/dirtbags/pcapdb
[73] https://github.com/google/stenographer
[74] https://play.google.com/store/apps/details?id=app.greyshirts.sslcapture
[75] https://cloudshark.io
[76] https://www.pcapr.net
[77] https://www.colasoft.com/packet_builder/
[78] http://www.hping.org
[79] https://scapy.net
[80] http://netdude.sourceforge.net

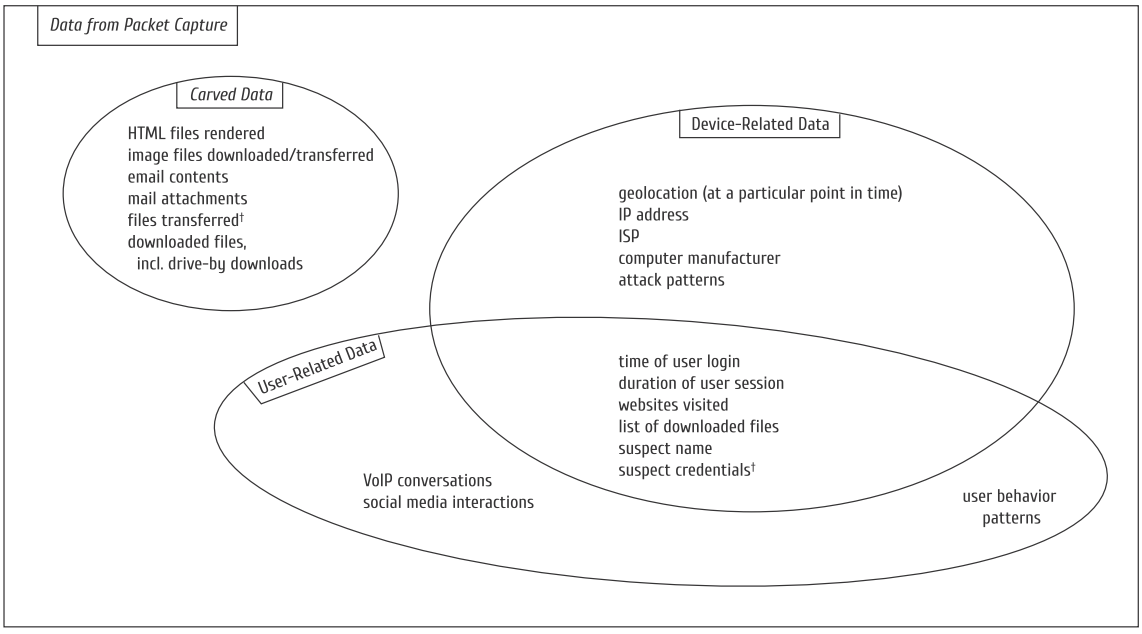


**Fig. 5.** Primary evidence types derived from network packets for forensic investigations.

### 5.4. Comparisons of packet analyzers for forensic applications

The primary use of packet analyzers in network forensics can be categorized by the data that can be extracted or reconstructed from packet data, and by the level of network activity that can be traced back.

#### 5.4.1. Reconstruction and carving capabilities

Carving can provide both direct and indirect forensic evidence of various nature (see Fig. 5).

Purpose-built carvers, such as *tcpflow* (Garfinkel, 2013), the *Packet Capture Forensic Evidence eXtractor (pcapfex)*,[81] and *File-TSAR*,[82] can efficiently extract files form packet capture. *NetScout* TruView[83] is a high-performance stream-to-disk packet sniffer for tracing abnormal user behavior. It provides advanced filters to find patterns in user behavior, and to identify tampering and compliance and security violations. Its *ClearSight Analyzer* can play back FTP traffic, messaging, email correspondence, and voice and video calls to quickly extract digital evidence. *NetworkMiner*[84] acts as a passive network sniffer, which can detect operating systems, sessions, hostnames, open ports, etc. without putting any traffic on the network; as a packet analyzer that parses pcap files; and as a network carver that reassembles transmitted files. *Cutter* is a tool for the forensic analysis of SCADA network traffic (Senthivel et al., 2017). It can identify transfers of logic programs and configuration files to/from a PLC in a network packet capture, and extract them for analysis.

#### 5.4.2. Tracing capabilities

In forensic applications, the logical grouping of packet-related data determines what level of context can be obtained from a packet capture. Packet analyzers may trace one or more levels of network information, i.e., packet data, packet metadata, flow, session, client-server communication, and payload, and many specialize in some of these (Lovanshi and Bansal, 2019). Although related information might be obtained from packet capture files by many analyzers, some, such as Tranalyzer and TCPflow, are better at flow analysis, while others, such as Fragroute and NetScout, are more suitable when user sessions are analyzed, as shown in Fig. 6.

CoralReef is the best choice when autonomous systems have to be identified, and ngrep when patterns or regular expressions have to be matched in the data payload of packets.

## 6. Research challenges and future directions in packet analysis

Utilizing machine learning in packet analysis is evolving into a complex research field that aimes at addressing the analysis of unknown features and encrypted network data streams (Yin et al., 2018), packet analysis in software-defined networks (Indira et al., 2019), and so on. In fact, machine learning-based approaches have a potential in addressing some of the issues of packet analysis in regards to big network data (Yoon and DeBiase, 2018), which affects more and more packet sniffing implementations.

The packet analysis of Internet of Things (IoT) networks plays an increasingly important role in fighting cyber-crime and mass surveillance. For example, IoT packet analysis can help detect distributed denial-of-service (DDoS) attacks (Salim et al., 2019) and the process of botnet forming (Kumar and Lim, 2020).

With the increasing share of cloud-based services, there is a growing need for performing packet capturing and analysis in cloud environments rather than using packet capture files of network segments. The government and financial sectors, cyber-defence and security applications, cloud-managed services, VoIP services, etc. utilize cloud storage and cloud computing services, with additional complexities on top of the source and destination IPs, protocols, and port numbers. For example, this is the very

[81] https://github.com/vikwin/pcapfex
[82] https://polytechnic.purdue.edu/facilities/cybersecurity-forensics-lab/tools
[83] https://www.netscout.com/sites/default/files/2018-09/6002359_TruView_Brochure.pdf
[84] https://www.netresec.com/?page=NetworkMiner

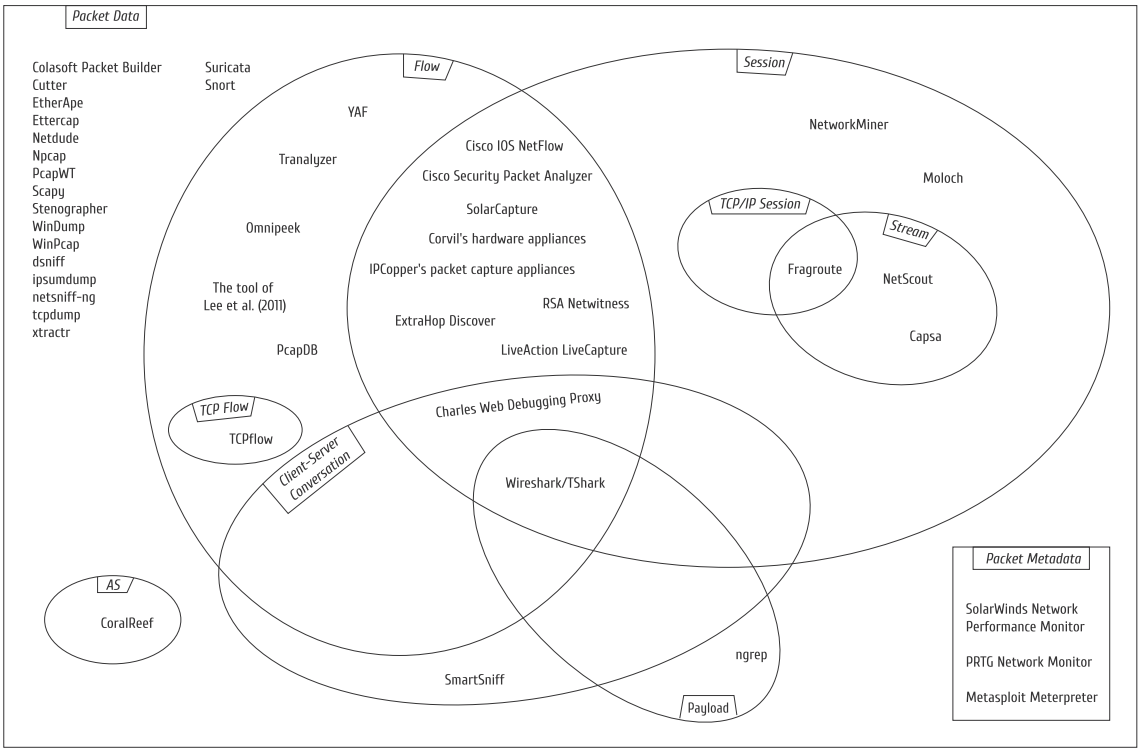


**Fig. 6.** Packet analyzers by the suitability for analyzing at a particular level of network information context.

reason for Amazon introducing *virtual private cloud (VPC) traffic mirroring*, which allows capturing and inspecting AWS network traffic at scale. This is done by selecting a network resource's (such as an EC2's) network interface and an elastic network interface or load balancer on another EC2 instance. So for example, if the network resource in a setup is an EC2 resource, and the EC2 mirror target runs tcpdump, the traffic being sent to the mirror destination is encapsulated using a VXLAN interface on the destination of the mirror.

Achieving a favorable tradeoff between privacy and packet analysis has long been a challenge (Yurcik et al., 2008), and urges research efforts in the area of privacy-preserving deep packet inspection (Li et al., 2017). The legal issues and concerns about invading privacy with packet analysis of wireless network traffic (Ohm, 2014) and IoT devices (Vukojević, 2015) continue to grow, which need more research. The corresponding technologies have to comply with the growing number of national and international policies and interception laws.

## 7. Conclusions

Analyzing network packets is fundamental in network forensics to collect data needed to obtain a clear understanding of online user actions happened at a particular point in time, and to serve evidence admissible in court. Even though some might be skeptical about the trustworthiness of information retrieved or reconstructed from packet data, network packets complement other information, such as corporate firewall logs or CCTV footage, and in many cases they form the one and only information source about what has happened during, and who was involved in, an online activity. Because employing packet analysis in network forensics differs from other application areas, such as intrusion detection, the potential of packets in providing forensic evidence has been explained and the limitations highlighted.

Both hardware and software implementations are available for packet sniffing, however, the capabilities of these tools vary greatly in terms of properties, supported protocols, interface, and licensing. This paper presented comparisons of state-of-the-art packet analyzers from multiple viewpoints.

The comprehensive review presented in this paper helps the reader gain a solid understanding of the processes involved in, and the tools designed for, packet analysis, along with the specific requirements of network forensics. This can be used for designing novel algorithms as well as innovative tools and methods for packet analysis in forensic applications.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

Afanasyev, M., Kohno, T., Ma, J., Murphy, N., Savage, S., Snoeren, A.C., Voelker, G.M., 2011. Privacy-preserving network forensics. Commun. ACM 54 (5), 78–87. https://doi.org/10.1145/1941487.1941508.

Agrawal, N., Tapaswi, S., 2017. The performance analysis of honeypot based intrusion detection system for wireless network. Int. J. Wirel. Inf. Netw. 24 (1), 14–26. https://doi.org/10.1007/s10776-016-0330-3.

Al-Duwairi, B., Govindarasu, M., 2006. Novel hybrid schemes employing packet marking and logging for IP traceback. IEEE T. Parall. Distr. 17 (5), 403–418. https://doi.org/10.1109/TPDS.2006.63.

Alhawi, O.M.K., Baldwin, J., Dehghantanha, A., 2018. Leveraging machine learning techniques for Windows ransomware network traffic detection. In: Dehghantanha, A., Conti, M., Dargahi, T. (Eds.), Cyber Threat Intelligence. Springer, Cham, pp. 93–106. https://doi.org/10.1007/978-3-319-73951-9_5.

Alshammari, R., Zincir-Heywood, A.N., 2015. Identification of VoIP encrypted traffic using a machine learning approach. J. King Saud Univ. Comput. Inf. Sci. 27 (1), 77–92. https://doi.org/10.1016/j.jksuci.2014.03.013.

Alsmadi, I., Burdwell, R., Aleroud, A., Wahbeh, A., Al-Qudah, M., Al-Omari, A., 2018. Network forensics: lesson plans. Practical Information Security: A Competency-Based Education Course. Springer, Cham, pp. 245–282. https://doi.org/10.1007/

978-3-319-72119-4_11.
Ansari, S., Rajeev, S.G., Chandrashekar, H.S., 2003. Packet sniffing: a brief introduction. IEEE Potentials 21 (5), 17–19. https://doi.org/10.1109/MP.2002.1166620.
Bellovin, S.M., Leech, M., 2000. ICMP traceback messages. https://www.ietf.org/proceedings/51/I-D/draft-ietf-itrace-00.txt.
Ben-Asher, N., Oltramari, A., Erbacher, R.F., Gonzalez, C., 2015. Ontology-based adaptive systems of cyber defense. In: Laskey, K.B., Emmons, I., Costa, P.C.G., Oltramari, A. (Eds.), Proceedings of the Semantic Technology for Intelligence, Defense, and Security. RWTH Aachen, Aachen, pp. 34–41. http://ceur-ws.org/Vol-1523/STIDS_2015_T05_BenAsher_etal.pdf.
Beverly, R., Garfinkel, S., Cardwell, G., 2011. Forensic carving of network packets and associated data structures. Digit. Invest. 8, S78–S89. https://doi.org/10.1016/j.diin.2011.05.010.
Bhandari, A., Gautam, S., Koirala, T.K., Islam, M.R., 2017. Packet sniffing and network traffic analysis using TCP—a new approach. In: Kalam, A., Das, S., Sharma, K. (Eds.), Advances in Electronics, Communication and Computing. Springer, Singapore, pp. 273–280. https://doi.org/10.1007/978-981-10-4765-7_28.
Boukhtouta, A., Mokhov, S.A., Lakhdari, N.-E., Debbabi, M., Paquet, J., 2016. Network malware classification comparison using DPI and flow packet headers. J. Comput. Virol. Hacking Tech. 12 (2), 69–100. https://doi.org/10.1007/s11416-015-0247-x.
Broadway, J., Turnbull, B., Slay, J., 2008. Improving the analysis of lawfully intercepted network packet data captured for forensic analysis. In: Jakoubi, S., Tjoa, S., Weippl, E.R. (Eds.), Third International Conference on Availability, Reliability and Security. IEEE Computer Society, Los Alamitos, CA, USA, pp. 1361–1368. https://doi.org/10.1109/ARES.2008.122.
Burch, H., Cheswick, B., 2000. Tracing anonymous packets to their approximate source. Proceedings of the 14th USENIX Conference on System Administration. USENIX, Berkeley, CA, USA, pp. 319–328. https://www.usenix.org/legacy/publications/library/proceedings/lisa2000/full_papers/burch/burch_html/.
Burschka, S., Dupasquier, B., 2016. Tranalyzer: versatile high performance network traffic analyser. 2016 IEEE Symposium Series on Computational Intelligence. IEEE, Piscataway, NJ, USA. https://doi.org/10.1109/SSCI.2016.7849909.
Carvalho, D.A., Pereira, M., Freire, M.M., 2009. Towards the detection of encrypted BitTorrent traffic through deep packet inspection. In: Ślęzak, D., Kim, T.-H., Fang, W.-C., Arnett, K.P. (Eds.), Security Technology. Springer, Heidelberg, pp. 265–272. https://doi.org/10.1007/978-3-642-10847-1_33.
Chapman, C., 2016. Using Wireshark and TCP dump to visualize traffic. In: Network Performance and Security: Testing and Analyzing Using Open Source and Low-Cost Tools. Syngress, Cambridge, MA, USA. https://doi.org/10.1016/B978-0-12-803584-9.00007-X.
Clarke, N., Li, F., Furnell, S., 2017. A novel privacy preserving user identification approach for network traffic. Comput. Secur. 70, 335–350. https://doi.org/10.1016/j.cose.2017.06.012.
Cui, Y., Xue, J., Wang, Y., Liu, Z., Zhang, J., 2018. Research of Snort rule extension and APT detection based on APT network behavior analysis. In: Zhang, H., Zhao, B., Yan, F. (Eds.), Trusted Computing and Information Security. Springer, Singapore, pp. 51–64. https://doi.org/10.1007/978-981-13-5913-2_4.
Das, R., Tuna, G., 2017. Packet tracing and analysis of network cameras with Wireshark. In: Genge, B., Haller, P. (Eds.), 5th International Symposium on Digital Forensic and Security. IEEE, Piscataway, NJ, USA. https://doi.org/10.1109/ISDFS.2017.7916510.
Dong, S., Jain, R., 2019. Flow online identification method for the encrypted Skype. J. Netw. Comput. Appl. 132, 75–85. https://doi.org/10.1016/j.jnca.2019.01.007.
Duncan, R., Jungck, P., 2009. packetC language for high performance packet processing. 11th IEEE International Conference on High Performance Computing and Communications. IEEE Computer Society, Los Alamitos, CA, USA, pp. 450–457. https://doi.org/10.1109/HPCC.2009.89.
Garfinkel, S.L., 2013. Passive TCP Reconstruction and Forensic Analysis with Tcpflow. Technical Report. Naval Postgraduate School. https://core.ac.uk/download/pdf/36728558.pdf.
Gong, C., Sarac, K., 2005. IP traceback based on packet marking and logging. IEEE International Conference on Communications. IEEE, Piscataway, NJ, USA, pp. 1043–1047. https://doi.org/10.1109/ICC.2005.1494507.
Goyal, P., Goyal, A., 2017. Comparative study of two most popular packet sniffing tools-Tcpdump and Wireshark. 9th International Conference on Computational Intelligence and Communication Networks. IEEE, pp. 77–81. https://doi.org/10.1109/CICN.2017.8319360.
Guo, Y., Gao, Y., Wang, Y., Qin, M., Pu, Y., Wang, Z., Liu, D., Chen, X., Gao, T., Lv, T., Fu, Z., 2017. DPI & DFI: a malicious behavior detection method combining deep packet inspection and deep flow inspection. Procedia Engineer. 174, 1309–1314. https://doi.org/10.1016/j.proeng.2017.01.276.
Hong, X., Hu, C., Wang, Z., Wang, G., Wan, Y., 2012. VisSRA: visualizing Snort rules and alerts. In: Tomar, G.S., Sharma, T.N., Bhatnagar, D. (Eds.), Fourth International Conference on Computational Intelligence and Communication Networks. IEEE Computer Society, Los Alamitos, CA, USA, pp. 441–444. https://doi.org/10.1109/CICN.2012.207.
Huang, J., Zhu, B., Chen, Z., 2012. Video traffic detection method for deep packet inspection. In: Jin, D., Lin, S. (Eds.), Advances in Computer Science and Information Engineering, 2. Springer, Heidelberg, pp. 135–140. https://doi.org/10.1007/978-3-642-30223-7_22.
Hurd, D., 2018. Endace fusion partners: redefining cybersecurity with Cisco. https://youtu.be/iRagH8y0GBA.
Indira, B., Valarmathi, K., Devaraj, D., 2019. An approach to enhance packet classification performance of software-defined network using deep learning. Soft Comput. 23 (18), 8609–8619. https://doi.org/10.1007/s00500-019-03975-8.
Islam, M.R., Koirala, T.K., Khatun, F., 2018. Network traffic analysis and packet sniffing using UDP. In: Bera, R., Sarkar, S.K., Chakraborty, S. (Eds.), Advances in Communication, Devices and Networking. Springer, Singapore, pp. 907–914. https://doi.org/10.1007/978-981-10-7901-6_97.
Jandaeng, C., 2016. Embedded packet logger for network monitoring system. In: Sulaiman, H.A., Othman, M.A., Othman, M.F.I., Rahim, Y.A., Pee, N.C. (Eds.), Advanced Computer and Communication Engineering Technology. Springer, Cham, pp. 1093–1102. https://doi.org/10.1007/978-3-319-24584-3_93.
Johansen, G., 2017. Acquiring host-based evidence. In: Digital Forensics and Incident Response: an Intelligent Way to Respond to Attacks. Packt Publishing, Birmingham, UK.
Joshi, R., Pilli, E.S., 2016. Network forensic tools. In: Fundamentals of Network Forensics. Springer, London, pp. 71-93.
Jungck, P., Duncan, R., Mulcahy, D., 2011. packetC Programming. Apress. https://doi.org/10.1007/978-1-4302-4159-1.
Kaushik, A.K., Pilli, E.S., Joshi, R.C., 2010. Network forensic analysis by correlation of attacks with network attributes. In: Das, V.V., Vijaykumar, R. (Eds.), Information and Communication Technologies. Springer, Heidelberg, pp. 124–128. https://doi.org/10.1007/978-3-642-15766-0_18.
Kim, H.S., Kim, H.K., 2011. Network forensic evidence acquisition (NFEA) with packet marking. In: Ninth International Symposium on Parallel and Distributed Processing with Applications Workshops. IEEE Computer Society, Los Alamitos, CA, USA, pp. 388–393. https://doi.org/10.1109/ISPAW.2011.27.
Kim, H., Kim, E., Kang, S., Kim, H.K., 2015. Network forensic evidence generation and verification scheme (NFEGVS). Telecommun. Syst. 60 (2), 261–273. https://doi.org/10.1007/s11235-015-0028-3.
Kim, Y.-H., Konow, R., Dujovne, D., Turletti, T., Dabbous, W., Navarro, G., 2015. PcapWT: an efficient packet extraction tool for large volume network traces. Comput. Network. 79, 91–102. https://doi.org/10.1016/j.comnet.2014.12.007.
Kumar, A., Lim, T.J., 2020. Early detection of Mirai-like IoT bots in large-scale networks through sub-sampled packet traffic analysis. In: Arai, K., Bhatia, R. (Eds.), Advances in Information and Communication. Springer, Cham, pp. 847–867. https://doi.org/10.1007/978-3-030-12385-7_58.
Lee, Y., Kang, W., Lee, Y., 2011. A Hadoop-based packet trace processing tool. In: Domingo-Pascual, J., Shavitt, Y., Uhlig, S. (Eds.), Traffic Monitoring and Analysis. Springer, Heidelberg, pp. 51–63. https://doi.org/10.1007/978-3-642-20305-3_5.
Lee, C., Park, M., Lee, J., Joe, I., 2012. Design and implementation of packet analyzer for IEC 61850 communication networks in smart grid. In: Kim, T., Ko, D., Vasilakos, T., Stoica, A., Abawajy, J. (Eds.), Computer Applications for Communication, Networking, and Digital Contents. Springer, Heidelberg, pp. 33–40. https://doi.org/10.1007/978-3-642-35594-3_5.
Li, J., Su, J., Wang, X., Sun, H., Chen, S., 2017. CloudDPI: cloud-based privacy-preserving deep packet inspection via reversible sketch. In: Wen, S., Wu, W., Castiglione, A. (Eds.), Cyberspace Safety and Security. Springer, Cham, pp. 119–134. https://doi.org/10.1007/978-3-319-69471-9_9.
Lotfollahi, M., Siavoshani, M.J., Zade, R.S.H., Saberian, M., 2019. Deep Packet: a novel approach for encrypted traffic classification using deep learning. Soft Comput. https://doi.org/10.1007/s00500-019-04030-2.
Lovanshi, M., Bansal, P., 2019. Comparative study of digital forensic tools. In: Shukla, R.K., Agrawal, J., Sharma, S., Tomer, G.S. (Eds.), Data, Engineering and Applications. Springer, Singapore, pp. 195–204. https://doi.org/10.1007/978-981-13-6351-1_15.
Manesh, T., Brijith, B., Singh, M.P., 2011. An improved approach towards network forensic investigation of HTTP and FTP protocols. In: Nagamalai, D., Renault, E., Dhanuskodi, M. (Eds.), Advances in Parallel Distributed Computing. Springer, Heidelberg, pp. 385–392. https://doi.org/10.1007/978-3-642-24037-9_38.
Mielczarek, W., Moń, T., 2015. USB data capture and analysis in Windows using USBPcap and Wireshark. In: Gaj, P., Kwiecień, A., Stera, P. (Eds.), Computer Networks. Springer, Cham, pp. 431–443. https://doi.org/10.1007/978-3-319-19419-6_41.
Murugesan, V., Selvaraj, M.S., Yang, M.-H., 2018. HPSIPT: a high-precision single-packet IP traceback scheme. Comput. Network. 143, 275–288. https://doi.org/10.1016/j.comnet.2018.07.013.
Ndatinya, V., Xiao, Z., Manepalli, V.R., Meng, K., Xiao, Y., 2015. Network forensics analysis using Wireshark. Int. J. Secur. Netw. 10 (2), 91–106. https://doi.org/10.1504/IJSN.2015.070421.
Nikkel, B.J., 2005. Generalizing sources of live network evidence. Digit. Invest. 2 (3), 193–200. https://doi.org/10.1016/j.diin.2005.08.001.
Ning, J., Pelechrinis, K., Krishnamurthy, S.V., Govindan, R., 2013. On the trade-offs between collecting packet level forensic evidence and data delivery performance in wireless networks. In: Kim, D.-I., Mueller, P. (Eds.), 2013 IEEE International Conference on Communications. IEEE, Piscataway, NJ, USA, pp. 1688–1693. https://doi.org/10.1109/ICC.2013.6654760.
Ohm, P., 2014. Should sniffing Wi-Fi be illegal? IEEE Secur. Priv. 12 (1), 73–76. https://doi.org/10.1109/MSP.2014.14.
Orebaugh, A., Ramirez, G., Burke, J., Pesce, L., Wright, J., Morris, G., 2006. Wireshark & Ethereal Network Protocol Analyzer Toolkit. Syngress, Rockland, MA, USA. https://www.sciencedirect.com/book/9781597490733/.
Parra, G.L.T., Rad, P., Choo, K.-K.R., 2019. Implementation of deep packet inspection in smart grids and industrial Internet of Things: challenges and opportunities. J. Netw. Comput. Appl. 135, 32–46. https://doi.org/10.1016/j.jnca.2019.02.022.
Parvat, T.J., Chandra, P., 2015. A novel approach to deep packet inspection for intrusion detection. Procedia Comput. Sci. 45, 506–513. https://doi.org/10.1016/j.procs.2015.03.091.

Rahman, M., Khalib, Z.I.A., Ahmad, R.B., 2009. Performance evaluation of PNtMS: a portable network traffic monitoring system on embedded Linux platform. In: Zhou, J., Zhou, X. (Eds.), 2009 International Conference on Computer Engineering and Technology, II. IEEE Computer Society, Los Alamitos, CA, USA, pp. 108–113. https://doi.org/10.1109/ICCET.2009.37.

Richter, P., Wohlfart, F., Vallina-Rodriguez, N., Allman, M., Bush, R., Feldmann, A., Kreibich, C., Weaver, N., Paxson, V., 2016. A multi-perspective analysis of carrier-grade NAT deployment. In: Proceedings of the 2016 Internet Measurement Conference. ACM, New York, pp. 215–229. https://doi.org/10.1145/2987443.2987474.

Pimenta Rodrigues, G.A., De Oliveira Albuquerque, R., Gomes de Deus, F.E., De Sousa Jr., R.T., De Oliveira Júnior, G.A., García Villalba, L.J., Kim, T.-H., 2017. Cybersecurity and network forensics: analysis of malicious traffic towards a honeynet with deep packet inspection. Appl. Sci. 7 (10), 1082–1110. https://doi.org/10.3390/app7101082.

Rounsavall, R., 2017. Full network traffic capture and replay. In: Vacca, J.R. (Ed.), Computer and Information Security Handbook, third ed. Morgan Kaufmann, Cambridge, MA, USA. https://doi.org/10.1016/B978-0-12-803843-7.00062-4.

Salim, M.M., Rathore, S., Park, J.H., 2019. Distributed denial of service attacks and its defenses in IoT: a survey. J. Supercomput. https://doi.org/10.1007/s11227-019-02945-z.

Sanders, C., 2017. Practical Packet Analysis: Using Wireshark to Solve Real-World Network Problems. No Starch Press, San Francisco.

Savage, S., Wetherall, D., Karlin, A., Anderson, T., 2001. Network support for IP traceback. IEEE ACM Trans. Netw. 9 (3), 226-237.

Senthivel, S., Ahmed, I., Roussev, V., 2017. SCADA network forensics of the PCCC protocol. Digit. Invest. 22, S57–S65. https://doi.org/10.1016/j.diin.2017.06.012.

Shah, S.A.R., Issac, B., 2018. Performance comparison of intrusion detection systems and application of machine learning to Snort system. Future Gener. Comput. Syst. 80, 157–170. https://doi.org/10.1016/j.future.2017.10.016.

Shimonski, R., 2013. The Wireshark Field Guide. Syngress. https://doi.org/10.1016/C2012-0-07287-0.

Sikos, L.F. (Ed.), 2018. AI in Cybersecurity. Springer, Cham. https://doi.org/10.1007/978-3-319-98842-9.

Sikos, L.F., 2019. Knowledge representation to support partially automated honeypot analysis based on Wireshark packet capture files. In: Czarnowski, I., Howlett, R.J., Jain, L.C. (Eds.), Intelligent Decision Technologies 2019. Springer, Singapore, pp. 345–351. https://doi.org/10.1007/978-981-13-8311-3_30.

Snoeren, A.C., Partridge, C., Sanchez, L.A., Jones, C.E., Tchakountio, F., Kent, S.T., Strayer, W.T., 2001. Hash-based IP traceback. In: SIGCOMM '01. ACM. https://doi.org/10.1145/383059.383060.

Snoeren, A.C., Partridge, C., Sanchez, L.A., Jones, C.E., Tchakountio, F., Schwartz, B., Kent, S.T., Strayer, W.T., 2002. Single-packet IP traceback. IEEE/ACM Trans. Netw. 10 (6), 721–734. https://doi.org/10.1109/TNET.2002.804827.

Song, D.X., Perrig, A., 2001. Advanced and authenticated marking schemes for IP traceback. In: Proceedings of IEEE INFOCOM 2001, 3. IEEE, Piscataway, NJ, USA, pp. 878–886. https://doi.org/10.1109/INFCOM.2001.916279.

Stalla-Bourdillon, S., Papadaki, E., Chown, T., 2014. From porn to cybersecurity passing by copyright: how mass surveillance technologies are gaining legitimacy … the case of deep packet inspection technologies. Comput. Law Secur. Rep. 30 (6), 670–686. https://doi.org/10.1016/j.clsr.2014.09.006.

Stallings, W., Case, T.L., 2012. Business Data Communications: Infrastructure, Networking and Security. Pearson, Upper Saddle River, NJ, USA.

Stergiopoulos, G., Talavari, A., Bitsikas, E., Gritzalis, D., 2018. Automatic detection of various malicious traffic using side channel features on TCP packets. In: Lopez, J., Zhou, J., Soriano, M. (Eds.), Computer Security. Springer, Cham, pp. 346-362.

Stone, R., 2000. CenterTrack: an IP overlay network for tracking DoS floods. In: Proceedings of the 9th USENIX Security Symposium. USENIX, Berkeley, CA, USA, pp. 199–212. https://www.usenix.org/legacy/events/sec2000/full_papers/stone/stone.pdf.

Sy, D., Bao, L., 2006. CAPTRA: coordinated packet traceback. In: 5th International Conference on Information Processing in Sensor Networks. ACM, New York, pp. 152–159. https://doi.org/10.1145/1127777.1127803.

Thomas, B., Mullins, B., Peterson, G., Mills, R., 2011. An FPGA system for detecting malicious DNS network traffic. In: Peterson, G., Shenoi, S. (Eds.), Advances in Digital Forensics VII. Springer, Heidelberg, pp. 195–207. https://doi.org/10.1007/978-3-642-24212-0_15.

Turnbull, B., Slay, J., 2007. Wireless forensic analysis tools for use in the electronic evidence collection process. In: Ralph, H., Sprague, J. (Eds.), Proceedings of the 40th Annual Hawaii International Conference on System Sciences. IEEE Computer Society, Los Alamitos, CA, USA. https://doi.org/10.1109/HICSS.2007.617.

van de Wiel, E., Scanlon, M., Le-Khac, N.-A., 2018. Enabling non-expert analysis of large volumes of intercepted network traffic. In: Peterson, G., Shenoi, S. (Eds.), Advances in Digital Forensics XIV. Springer, Cham, pp. 183–197. https://doi.org/10.1007/978-3-319-99277-8_11.

Vukojević, S., 2015. Violation of user privacy by IPTV packet sniffing in home network. In: Biljanovic, P., Butkovic, Z., Skala, K., Mikac, B., Cicin-Sain, M., Sruk, V., Ribaric, S., Gros, S., Vrdoljak, B., Mauher, M., Sokolic, A. (Eds.), 38th International Convention on Information and Communication Technology, Electronics and Microelectronics. IEEE, pp. 1338–1343. https://doi.org/10.1109/MIPRO.2015.7160482.

Wang, M.-H., Yu, C.-M., Lin, C.-L., Tseng, C.-C., Yen, L.-H., 2014. KPAT: a kernel and protocol analysis tool for embedded networking devices. In: Jamalipour, A., Deng, D.-J. (Eds.), 2014 IEEE International Conference on Communications. IEEE, Piscataway, NJ, USA, pp. 1160–1165. https://doi.org/10.1109/ICC.2014.6883478.

Xiang, Y., Zhou, W., Guo, M., 2008. Flexible deterministic packet marking: an IP traceback system to find the real source of attacks. IEEE T. Parall. Distr. 20 (4), 567–580. https://doi.org/10.1109/TPDS.2008.132.

Yang, J., Zhang, Y., King, R., Tolbert, T., 2018. Sniffing and chaffing network traffic in stepping-stone intrusion detection. In: Barolli, L., Takizawa, M., Enokido, T., Ogiela, M.R., Ogiela, L., Javaid, N. (Eds.), 32nd International Conference on Advanced Information Networking and Applications Workshops. IEEE Computer Society, Los Alamitos, CA, USA, pp. 515–520. https://doi.org/10.1109/WAINA.2018.00137.

Yin, C., Wang, H., Wang, J., 2018. Network data stream classification by deep packet inspection and machine learning. In: Park, J.J., Loia, V., Choo, K.-K.R., Yi, G. (Eds.), Advanced Multimedia and Ubiquitous Engineering. Springer, Singapore, pp. 245–251. https://doi.org/10.1007/978-981-13-1328-8_31.

Yin, C., Wang, H., Yin, X., Sun, R., Wang, J., 2018. Improved deep packet inspection in data stream detection. J. Supercomput. 75 (8), 4295–4308. https://doi.org/10.1007/s11227-018-2685-y.

Yoon, J., DeBiase, M., 2018. Real-time analysis of big network packet streams by learning the likelihood of trusted sequences. In: Chin, F.Y.L., Chen, C.L.P., Khan, L., Lee, K., Zhang, L.-J. (Eds.), Big Data – BigData 2018. Springer, Cham, pp. 43–56. https://doi.org/10.1007/978-3-319-94301-5_4.

Yu, C., Lan, J., Xie, J., Hu, Y., 2018. QoS-aware traffic classification architecture using machine learning and deep packet inspection in SDNs. Procedia Comput. Sci. 131, 1209–1216. https://doi.org/10.1016/j.procs.2018.04.331.

Yurcik, W., Woolam, C., Hellings, G., Khan, L., Thuraisingham, B., 2008. Making quantitative measurements of privacy/analysis tradeoffs inherent to packet trace anonymization. In: Tsudik, G. (Ed.), Financial Cryptography and Data Security. Springer, Heidelberg, pp. 323–324. https://doi.org/10.1007/978-3-540-85230-8_33.