International Journal of Scientific Research in Science, Engineering and Technology
Print ISSN: 2395-1990 | Online ISSN : 2394-4099 (www.ijsrset.com)
doi : https://doi.org/10.32628/IJSRSET

An Analytical Review on Packet Analysis for Network Forensics and Deep Packet Inspection in Network

Aniruddha R. Jaipurkar, Dr. Nilesh Marathe
Department of Computer Science and Engineering, Network & Security, MIT school of Engineering, Pune Maharashtra, India

Article Info
Volume 9, Issue 4

Page Number : 147-167

Publication Issue :
July-August-2022

Article History
Accepted : 05 July 2022
Published: 20 July 2022

ABSTRACT

Packet analysis is a fundamental traceback approach in network forensics. It can play back even the entirety of the network traffic for a specific point in time, provided that the packet details captured are sufficiently detailed. This can be utilised to discover evidence of malicious online behaviour, data breaches, unauthorised website access, malware infection, and attempted intrusions, as well as to reconstruct image files, documents, email attachments, and other types of data that have been transmitted across the network. This article offers a detailed study of the use of packet analysis in network forensics, including deep packet inspection. It also gives a discussion of AI-powered packet analysis algorithms that have enhanced network traffic classification and pattern identification capabilities. In light of the fact that not all information obtained through a network can be used as evidence in a legal proceeding, a comprehensive list of the kinds of digital information that might be allowed has been compiled. We take a look at the capabilities of both physical appliances and software packet analyzers from the point of view of their possible use in forensic investigations of computer networks.

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

I. INTRODUCTION

1. Introduction to packet analysis in network forensics
Because of the ever-growing popularity of online services, security professionals and law enforcement organisations are under increasing pressure to develop innovative strategies for investigating cybercrimes and locating evidence that may be presented in court. Large volumes of data are transferred through communication networks by online services in a variety of formats, the most common of which is the network packet. Online services transport this data in a variety of ways. According to Stallings and Case (2012), these are groupings of bits that incorporate data as well as control information. More specifically, this term is used to refer to a network layer (OSI Layer 3) protocol data unit. They are the smallest unit of data that may be intercepted and logged regarding network

traffic flow while it is moving across a packet-switched network1, and they comprise of control information (the source and destination IP address, error detection codes, and sequencing information) and payload (intended message). A set of bits that contains data along with one or more addresses and other protocol control information is referred to as a frame, and it is a data unit that is found in OSI Layer 2 (the data link layer) (Stallings and Case, 2012). The equivalent is referred to as a segment in OSI Layer 4, which is the transport layer (or datagram).

Network packets can be utilised in forensic investigations and may even produce evidence that is admissible in a court case if they are successfully collected, saved, and processed once they have been obtained from a network. Be aware that throughout this entire piece, we are going to use the term "packet analysis," regardless of whether the actual content is a frame, packet, data gramme, or session, unless it is stated otherwise. This is because the term "packet analysis" encompasses all of these different types of content.

2. Capturing and storing network packets

Protocols are defined as "mechanisms to identify and create connections, as well as formatting standards and conventions that are specified for data transfer," and they are what make it possible for network devices to communicate with one another. Using purpose-built software, it is possible to examine the data from a network and separate the traffic into its many components. Packet analyzers are the same thing as protocol analyzers, except that their primary purpose is to analyse packets (packet sniffers, sometimes network analyzers). By utilising a method known as "packet capturing," these software solutions are able to capture and record the network traffic that is moving over a digital network or a portion of a network. After then, the collected packets can be studied by decoding the raw data contained within the packets and visualised by showing various fields in order to interpret the content of the packets (Chapman, 2016). When a capable wired network interface controller (NIC) or wireless network interface controller (WNIC) is put into promiscuous mode, all of the network traffic that is received can be sent to the central processing unit (CPU), rather than just the frames that the controller is specifically programmed to receive. This is referred to as a full duplex mode. The Berkeley Packet Filter (BPF) is a programme that allows for the filtering of packets, such as receiving only those packets that begin a TCP connection. This programme is available on most Linux distributions. Because BPF only returns the packets that make it through the filter, there is no need for the operating system to copy irrelevant packets to the kernel in order for them to be processed. This results in a significant increase in the speed of the system. Extended BPF (eBPF), which is an improvement on the original BPF, allows for loops since it supports not only forward jumps but also backward hops in addition to the standard forward jumps. Aggregating event statistics can also be accomplished with eBPF through the utilisation of maps, which are global data repositories.

There are a few different ways to "tap into the wire," and the method that should be used to do so is determined by the networking environment in which the device whose traffic to be studied is situated. In the relatively uncommon networks that make use of hubs, a packet sniffer is able to view all of the devices that are connected to the network for the simple reason that traffic that is delivered through a hub is passed through every port that is attached to that hub. The visibility of a packet sniffer is restricted to the port into which it is plugged in a networking environment that uses switched connections. Port mirroring (also known as port spanning), hubbing out, employing a tap, and ARP cache poisoning (also known as ARP spoofing) are the four primary methods that can be used to capture traffic from a target device on switched networks. The answer to this question is dependent on the situation: The first one is only a possibility 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 it has an empty port into which we can plug our sniffer; the second one requires having physical access to the switch the target device is plugged into; the third one calls for a specialised piece of hardware known as a network tap to be connected to the network; and the fourth one necessitates having information about

Analyzing network packets, which include valuable information about network activities and which can be used to compile and report network statistics as well as troubleshoot client-server conversations, is helpful in both of these areas. Network packet capture files store a lot of information about online user activity that can be useful in network forensics. Some examples of this information include visited websites and the amount of time spent browsing them,2 successful and unsuccessful login attempts, credentials, illegal file downloads, intellectual property abuse, and other types of information. Not only do packet files contain a plethora of information, but data may also be obtained from them in a variety of groupings, such as individual frames, client-server talks, packet streams, flows, and sessions. In addition, packet files contain a richness of information. In the field of network forensics, packet analysis can be used to collect evidence for investigations of digital activities; it can also be used to detect malicious network traffic and behaviour, such as attempted intrusions and misuse of the network; it can also be used to identify man-in-the-middle attacks and malware, such as ransomware; and it can be used to identify security vulnerabilities (Alhawi et al., 2018).

The capture format that has become the de facto standard is called libpcap (pcap), and it is a binary format that supports timestamps with a nanosecond-precision.

Pcap files all have the general structure depicted in Fig. 1, despite the fact that this format changes slightly from implementation to implementation.

The maximum length of captured packets, measured in octets, the GMT offset, the timestamp precision, and the maximum length of captured packets are all included in the global header along with the magic number, which is used to identify the file format version and byte order. After this information, there may be zero, one, or more records containing the collected packet data. Each captured packet begins with the timestamp in seconds, the timestamp in microseconds, the number of octets of the packet that have been stored in file, and the actual length of the packet itself.

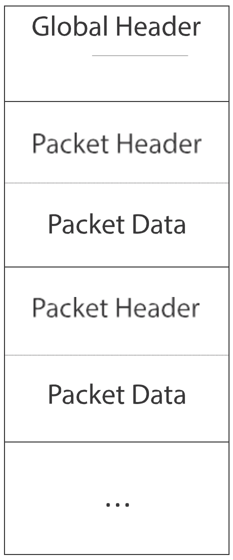


Figure 1 depicts the overall organisation of pcap files. The data packets only include the most recent N bytes of each individual packet, where N represents the length of the snapshot (typically smaller than 65,535).

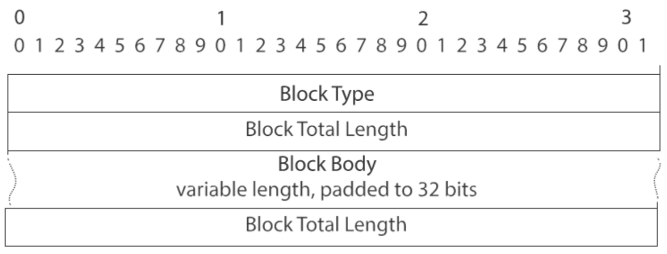
Next Generation Capture File Format (pcap) is the format that will replace pcap in the future (pcapng). Instead of merely being able to dump network packets, pcapng is capable of preserving a variety of data kinds with a generic block structure. This is in contrast to its previous limitation. The IETF is responsible for developing the structure of pcapng files, as may be seen in Figure 2.

The specifics of the block structure are determined by 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. Each of these block types has its own set of details regarding the block structure. The creation of further categories is now underway.

IETF RFC 1761.5 is where the format for the snoop capture is defined. Each snoop file is an array of octets that consists of a file header that has a fixed length and one or more packet records that have varying lengths. A 32-bit datalink type, a 32-bit version number, and a 64-bit identification pattern are all included in the header of each and every file.

The RedBack Smartedge pcap format, often known as SE400/800, was developed specifically for NetBSD running on PowerPC with intelligent packet-forwarding linecards. This format expands the pcap format with additional information regarding protocols and circuits. It is built on circuits, and its foundation is circuits.

Additional 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; the HP-UX nettl trace; the K12 text file format; the Microsoft Network Monitor (Net If the timestamp is added by the kernel or the CPU that the capture is offloaded to, or if a packet has been waiting in a ring buffer, then this precision is typically not available in practise because the packet capture implementation that is in place may only support milliseconds. This can occur in a number of scenarios, including when a packet has been waiting in a ring buffer.



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 recon- structed from network packet streams (network carving) (Beverly et al., 2011) using purpose-designed network carvers or packet ana- lyzers 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 prop- erties. 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, often known as DPI, is a sort of packet analysis that examines not only the information included in the packet header but also the information contained in the packet payload. DPI can be utilised to identify excessive levels of non-business traffic in enterprises, such as social media use, that require being 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 behaviour (Guo et al., 2017); malicious traffic (Stergiopoulos et al., 2018); intrusions (Parvat (Parra et al., 2019). In point of 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 United States),6 even if the sheer volume of traffic makes it impractical to record all traces of user activity. This is because 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 (Stalla-Bourdillon et al., 2014). Deep packet inspection is advantageous for network operators because it enables them to shape traffic and exert control over several forms of traffic, such as email, VoIP, and P2P. DPI services are provided by businesses such as NETSCOUT7 and Sandvine8. These services help prioritise network activity, enforce policies, and assist in the development of new service plans.

3.2. Using artificial intelligence in packet analysis

In network forensics, formal knowledge representation is used in the form of ontologies to automate the analysis of network packet sequences. This is done in order to reduce human error (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), have the ability to capture the semantics of actual network packets and provide terms to formally describe background knowledge in a form that a machine can understand. The datasets that make use of these definitions and codified expert knowledge, in conjunction with reasoning rules, may be utilised to infer new claims and make implicit network information explicit.

In their 2018 study, Shah and Issac used machine learning and developed a plugin in order to reduce the number of false positives that occurred during the detection of hostile traffic using Snort. This plugin decodes packet data, classifies network packets, differentiates between legal and malicious traffic, and reduces the probability of false positive alarms by using support vector machines (SVM), decision trees, a mix of SVM and fuzzy logic, and optimised SVM with firefly.

Deep packet inspection paired with semi-supervised machine learning is an effective method for efficiently categorising flows to recognise audio, video, and interactive data, which paves the way for fine-grained adaptive quality of service traffic engineering (Yu et al., 2018). The classifier is able to adjust to quickly shifting network traffic patterns through the use of periodic retraining with a dynamic flow database. This allows the classifier to adapt.

Deep Packet is the name of the deep learning-based methodology developed by Lotfollahi et al. (2019), which combines the processes of feature extraction and classification. Using two different types of deep neural network structures—stacked autoencoder (SAE) and convolutional neural network (CNN), respectively—it is able to categorise network traffic into classes such as FTP and P2P. Additionally, it is able to identify end-user applications such as Skype and BitTorrent. Not only can the Deep Packet method recognise encrypted traffic, but it can also differentiate between VPN and non-VPN network traffic. This is a significant advantage.

3.2.1. Optimizing and offloading packet processing

The use of implementation integrated circuits (ASICs), field programmable arrays (FPGAs), and graphics processing units are all viable options for achieving hardware acceleration and offloading in the context of network packet processing (GPUs). Offloading of most IPv4 and IPv6 traffic, IPsec VPN encryption, CAPWAP traffic, and multicast traffic are just a few of the types of traffic that the FortiGate's FortiASIC NP6 can handle.

3.3. Programming packet processing applications

Specialized hardware, often coded in Assembly or C, is required in order to carry out network packet analysis at speeds in the gigabits per second range, with deep packet inspection in particular requiring this level of processing power (Duncan and Jungck, 2009). Using a parallel packet processing paradigm in conjunction with the purpose-designed programming language packetC is an additional method that may be utilised. This language offers high-level constructs to define coding solutions for applications that include packet processing. It avoids type coercions or promotions to prevent unexpected data truncations or expansions, it supports a strong typing model with restrictive type casting to prevent unexpected side effects, and it simplifies and restricts type declarations in comparison to C. These features are intended to reduce the likelihood of unexpected type conflicts (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 evi- dence (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 informa- tion 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). 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 in- vestigations 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).

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 begin- ning 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 developed for a wide variety of applications and can be distinguished from one another based on their capacities and features, hardware resource utilisation, processing speed (Goyal and Goyal, 2017), supported protocols, user-friendliness, supported operating systems, supported network types, interface, licence, and type of implementation. There are a variety of packet analyzers available, and many of them enable both live capture and offline analysis. Only those network analyzers that handle hundreds of protocols are capable of doing a thorough inspection of individual packets as well as an analysis of a wide variety of forms of network traffic. Wireless analyzers, often known as WiFi analyzers, are a type of packet analyzer that can intercept traffic over wire-less networks. Some examples of wireless analyzers are Aircrack-ng,18 and Kismet. 19 There is a packet sniffer designed specifically for Bluetooth that goes by the name FTS4BT. 20

Data carving, capture file quality evaluation, anomaly detection, protocol encapsulation, and flexible packet aggregation are some of the features supported by some tools. The list of file types that may be opened by a packet analyzer might differ from one to the next, and some programmes can even decompress gzip files in real time. 21

The analyzers that come with a graphical user interface (GUI) feature typically have a packet browser that allows users to visualise the content of the packets, as well as a variety of display filters that show only the information that is pertinent to a specific task, as opposed to showing everything that was captured. Some packet analyzers are able to distinguish between different sorts of frames and visually represent this distinction using colour schemes.

The licencing structure of packet analyzers may be broken down into three categories: open source, freeware, and commercial. Types of licences that are commonly connected with Instrumentation for doing packet analysis

Distinguished examples of hardware packet analyzers include the Fluke Lanmeter series (which has since been 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 appli- ances,24 Corvil's hardware.

A few of these appliances are physical in nature (either embedded or dedicated), while others are bare-metal, virtualized, or hybrid in nature.

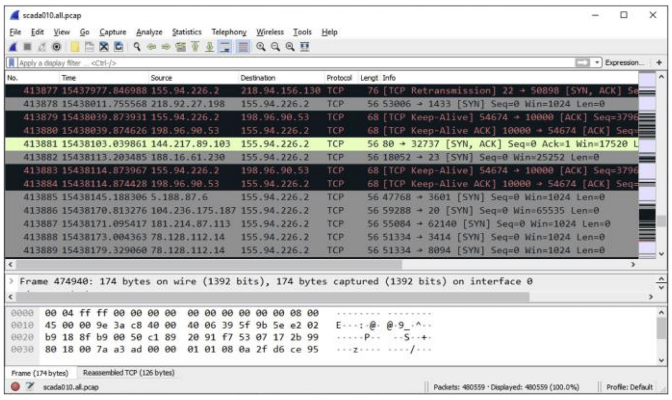
5.1. Packet analyzer software

There are purpose-designed packet analyzers and network tools that are included in the category of packet analyzer software tools. These tools include features for the collection and examination of packets. Intrusion detection software, proxies, vulnerability assessment tools, network scanners, and network monitoring tools are all examples of such types of network tools, all of which are utilised in the field of network forensics (Joshi and Pilli, 2016).

1997 saw the implementation of the Federal Bureau of Investigation's (FBI) configurable packet sniffer as a component of the system known as Carnivore (which was later renamed to DCS1000). It tracked users' activity on the internet, including their email communications. By 2005, it had been completely phased out. The free and open-source packet analyzer known as Ethereal was initially developed by Gerald Combs in 1998. In 2006, the programme was rebranded as Wireshark (Orebaugh et al., 2006). Wireshark has evolved over the years to become one of the most popular graphical packet capture and protocol analysis tools (Shimonski, 2013). It features an extremely user-friendly graphical user interface (GUI) for doing packet analysis (Sanders, 2017). This graphical user interface includes a customised packet browser that shows a maximum of three panes at the same time. These panes include a packet list, as well as the packet information and packet bytes of the packet that is presently chosen (see Fig. 3).

The programme utilises colouring rules to differentiate between inactive and active selected items, marked packets and ignored packets, follow streams (both client and server), and to display valid, invalid, and warning filters. Additionally, the programme is able to display marked packets and ignored packets. It also contains effective display filters that allow the user to zero in on the frames that are pertinent to a certain investigation, for example, by displaying just HTTPS traffic or network communication that is tied to a specific IP address. The capabilities of Wireshark may also be accessed using a command-line programme known as TShark, and Wireshark offers supplementary tools for the management of packet captures (capinfos, mergecap, editcap). Because of its flexible feature set, Wireshark is used extensively, and both industry professionals and academic researchers are concentrating their attention on it (Mielczarek and Mon, 2015; Das and Tuna, 2017; Alsmadi et al., 2018; Islam et al., 2018; Bhandari et al., 2017).

There are hardware appliances as well as software implementations that may be used for packet analysis; however, software tools are utilised far more frequently than hardware implementations. logging of individual IP packets on many networks. It has been demonstrated that Snort is effective for doing complicated network behaviour analysis, such as for the purpose of assisting in the detection of advanced persistent threats (APTs). Snort makes use of detection rules for a variety of network traffic types (Cui et al., 2018). In a manner analogous to that of Wireshark, development of Snort is ongoing, and a large number of third-party plugins are available to improve its functionality. One such plugin is VisSRA, which visualises Snort rules and alarms (Hong et al., 2012).


scada010.all.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.    Time             Source          Destination     Protocol Lengt Info
413877 15437977.846988 155.94.226.2    218.94.156.130  TCP      76 [TCP Retransmission] 22 → 50898 [SYN, ACK] Se
413878 15438011.755568 218.92.27.198   155.94.226.2    TCP      56 53006 → 1433 [SYN] Seq=0 Win=1024 Len=0
413879 15438039.873931 155.94.226.2    198.96.90.53    TCP      68 [TCP Keep-Alive] 54674 → 10000 [ACK] Seq=3796
413880 15438039.874626 198.96.90.53    155.94.226.2    TCP      68 [TCP Keep-Alive ACK] 10000 → 54674 [ACK] Seq=
413881 15438103.039861 144.217.89.103  155.94.226.2    TCP      56 80 → 32737 [SYN, ACK] Seq=0 Ack=1 Win=17520 L
413882 15438113.203485 188.16.61.230   155.94.226.2    TCP      56 18052 → 23 [SYN] Seq=0 Win=25252 Len=0
413883 15438114.873967 155.94.226.2    198.96.90.53    TCP      68 [TCP Keep-Alive] 54674 → 10000 [ACK] Seq=3796
413884 15438114.874428 198.96.90.53    155.94.226.2    TCP      68 [TCP Keep-Alive ACK] 10000 → 54674 [ACK] Seq=
413885 15438145.188306 5.188.87.6      155.94.226.2    TCP      56 47768 → 3601 [SYN] Seq=0 Win=1024 Len=0
413886 15438170.813276 104.236.175.187 155.94.226.2    TCP      56 59288 → 20 [SYN] Seq=0 Win=65535 Len=0
413887 15438171.095417 181.214.87.113  155.94.226.2    TCP      56 55084 → 62140 [SYN] Seq=0 Win=1024 Len=0
413888 15438173.004363 78.128.112.14   155.94.226.2    TCP      56 51334 → 3414 [SYN] Seq=0 Win=1024 Len=0
413889 15438179.329060 78.128.112.14   155.94.226.2    TCP      56 51334 → 8094 [SYN] Seq=0 Win=1024 Len=0

> Frame 474940: 174 bytes on wire (1392 bits), 174 bytes captured (1392 bits) on interface 0

0000  00 04 ff ff 00 00 00 00  00 00 00 00 00 00 08 00   ········ ········
0010  45 00 00 9e 3a c8 40 00  40 06 39 5f 9b 5e e2 02   E···:·@· @·9_·^··
0020  b9 18 8f b9 00 50 c1 89  20 91 f7 53 07 17 2b 99   ·····P··  ··S··+·
0030  80 18 00 7a a3 ad 00 00  01 01 08 0a 2f d6 ce 95   ···z···· ····/···

Frame (174 bytes)   Reassembled TCP (126 bytes)
scada010.all.pcap   Packets: 480559 · Displayed: 480559 (100.0%)   Profile: Default
```

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

Eddie Kohler, who was a student at MIT at the time, created ipsum- dump33 in 1999 with the intention of summarising TCP/IP dump files or other packet sources into a self-describing ASCII format that could be consumed by both humans and machines.

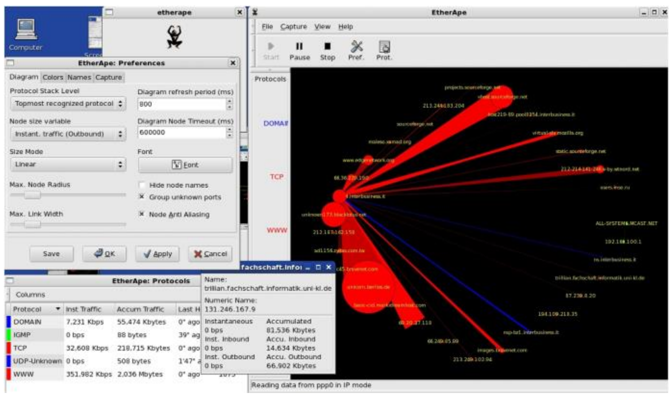
Dsniff34 was designed by Song for use on operating systems that are similar to Unix. It is a component of a larger suite of tools for network auditing and penetration testing. It decodes passwords that were communicated in cleartext across a switched or unswitched Ethernet network, for example, and it parses a variety of application protocols and extracts important information.

EtherApe, a tool for network traffic monitoring and packet sniffing, was created by Toledo in the year 2000 for the Unix operating system. This piece of software, which is both open source and free, can display the traffic on a network using graphs. In these graphs, each node represents a specific host, and the edges of the graphs indicate the connections between hosts. Different protocols may be distinguished from one another thanks to the use of color-coding for the nodes and connections in the network. As can be seen in Figure 4, the quantity of network traffic is graphically represented in a manner that is proportional to the breadth of the graph edges.

Tcpdump37 is a command-line utility that has been available for almost twenty years and is considered to be one of the de facto standard tools for dumping network packets and capturing them for further study. Additionally, it comes with a Windows implementation known as WinDump. 38 Tcpdump is created in tandem with libpcap,39 which is a well-known software library for capturing live network traffic data. This library is also used by packet analyzers and other applications having the capability of packet sniffing, such as Wireshark and Snort. The pcap Application Programming Interface is used by libpcap. The pcap application programming interface is also utilised by the packet sniffers WinPcap40 and Npcap41, as well as ngrep42, a programme that can locate regular expression matches inside the payloads of network packets.

Ettercap43 is a free and open source network security application that was created in 2001 by ALoR and NaGA to combat man-in-the-middle (MITM) attacks on local area networks (LANs). It shows the IP address and the Media Access Control address of any host that is connected to a network. It is able to identify hosts that have unauthorised IP addresses, and so it is able to detect attackers; but, it will not be able to detect an attacker who spoofs their IP address in order to utilise an allowed IP address (Agrawal and Tapaswi, 2017).

Karl von Randow created the Charles Web Debugging Proxy44 in 2002. It is an HTTP proxy, HTTP monitor, and reverse proxy that visualises all of the HTTP and SSL/HTTPS traffic that occurs between a computer and the Internet. The year after that, Eric Lawrence created Fiddler,45 a free online debugging proxy that can log HTTP and HTTPS traffic. Fiddler was developed by Eric Lawrence. It is possible to filter the data that was taken from the network in order to conceal sessions, emphasise the traffic that is of interest, bookmark breakpoints, and so on.



EtherApe: Preferences

Diagram | Colors | Names | Capture

Protocol Stack Level: Topmost recognized protocol
Diagram refresh period (ms): 800
Node size variable: Instant. traffic (Outbound)
Diagram Node Timeout (ms): 600000
Size Mode: Linear
Font: Font
Max. Node Radius
Max. Link Width
Hide node names
[✓ Group unknown ports]
[✓ Node Anti Aliasing]

Save | OK | Apply | Cancel

EtherApe: Protocols

Columns

Protocol	Inst Traffic	Accum Traffic	Last H
DOMAIN	7.231 Kbps	55.474 Kbytes	0" ago
IGMP	0 bps	88 bytes	39" ag
TCP	32.608 Kbps	218.715 Kbytes	0" ago
UDP-Unknown	0 bps	508 bytes	1'47" a
WWW	351.982 Kbps	2.036 Mbytes	0" ago

fachschaft.info

Name: trillian.fachschaft.informatik.uni-kl.de
Numeric Name: 131.246.167.9

Instantaneous	Accumulated
0 bps	81.536 Kbytes
Inst. Inbound	Accu. Inbound
0 bps	14.634 Kbytes
Inst. Outbound	Accu. Outbound
0 bps	66.902 Kbytes

Reading data from ppp0 in IP mode

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

The software includes a widget called the Session Inspector that has the capability to display the contents of a web session that has been recorded. This includes the status of the session, its headers, its caching, its cookies, its URLs, its protocols, the type of compression that was used, its re-directs, and so on.

2009 saw the release of Suricata46 by the Open Information Security Foundation. Suricata46 is an intrusion detection and prevention system (IDS/IPS) that is open source-based. Among its many capabilities, it can scan pcap files with IDS rulesets to find traces of suspicious or malicious network activities. In a manner analogous to that of Snort, Suricata is popular enough to have support for a multitude of third-party tools that may supplement it for the purposes of visualisation and analysis. These tools include Snorby,47 BASE, Sguil,48 u2platform (previously Aanval),49 and CERNE.50

During the same year, 2009, Daniel Borkmann created netsniff-ng, which is a free Linux network analyzer. It is a high-performance utility that makes advantage of zero-copy techniques for network packets. As a result, the Linux kernel does not need to transfer packets from kernel area to user space via system calls. The programme is able to collect, analyse, and playback raw 802.11 frames, in addition to supporting the standard pcap file format.

Finding vulnerabilities in online applications has never been easier than it is with WebScarab52, an integrated penetration testing tool that is user-friendly. It is confined to programmes that communicate using the HTTP and HTTPS protocols, although it does have the capability to analyse packets.

Tranalyzer53 is a piece of open-source software that may be used for network troubleshooting in addition to flow- and packet-based traffic analysis. It is developed on top of the libpcap library, and in addition to accepting IPv4 and IPv6 packets, it also takes Layer 2 and encapsulated packets, such as MPLS, L2TP, and GRE, from standard pcap files or live interfaces. Additionally, it accepts IPv4 and IPv6 packets from normal pcap files (Burschka and Dupasquier, 2016).

The logs that are created by tcpdump, snoop, EtherPeek, HP Net Metrix, and WinDump may be analysed with the help of a programme called tcptrace54. Yet Another Flowmeter (YAF)55 is a metre that measures the flow of information over a network. The metadata that YAF generates may be fed into the programme yafMeta2Pcap56 to generate pcap files specific to a given flow.

The SolarWinds Network Performance Monitor comes with a packet analyzer and also features deep packet inspection, which enables 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. Moreover, the packet analyzer58 comes standard with the

SolarWinds Network Performance Monitor57. The Paessler PRTG Network Monitor comes equipped with a large assortment of network monitoring functions, one of which is a packet capture tool known as the Packet Sniffer Sensor. This tool's capabilities for analysing packets are restricted to the examination of data packet headers; however, it does include additional functions, such as displaying how different kinds of network traffic make use of available bandwidth.

An Apache Hadoop-based packet processing tool was introduced by Lee et al. (2011) to address the inefficient processing of large packet capture files that is caused by traditional packet analyzers running on a single host with limited computing and storage resources. This tool can open even petabyte-sized packet capture files thanks to the MapReduce parallel processing paradigm. This application makes use of four innovative and representative calculation modules to calculate overall traffic statistics, periodic flow statistics, periodic simple statistics, and top N statistics respectively.

The packet analyzer developed by Lee et al. (2012) is specifically designed for rapid later access (FLA) and deep packet inspection of network packets by making use of IEC 61850 communication protocols. This analyzer was developed for communications between the server and client of substation automation systems (SASes). Each SAS consists of a station, a bay, and a process level. Considering the large number of packets that need to be processed in IEC 61850 networks, this analyzer was designed for these types of communications. When it comes to evaluating networks of this type, the authors say that their analyzer performs far better than other, more generic packet analyzers.

PcapWT is another programme that can quickly extract packets from massive network traces. It uses the wavelet tree data structure, which enables a rapid search and a good compression ratio. PcapWT's support for multi-threading results in improved performance while reading and writing random file data to and from solid-state drives (SSDs) (Kim et al., 2015).

The CoralReef software package developed by CAIDA is able to gather and analyse data in real time or from trace files obtained from passive Internet traffic monitors. The ability of CoralReef to classify packet traffic according to the source autonomous systems is one of its defining characteristics (ASes).

The programme known as Xtractr63 operates in a hybrid cloud environment and allows users to index, search, report on, extract, and collaborate on pcaps.

Cisco NetFlow64 compiles and organises statistical data on packets as they pass via various routing devices. It is able to determine the flows of packets for both incoming and outgoing IP packets. 65

The Capsa66 packet analyzer is an all-inclusive tool that offers support for more than 300 different protocols. It is able to display detailed information on packet decoding in hex, ASCII, and EBCDIC formats. It is able to reconstruct packet streams and make it possible for packet capture and analysis to be carried out automatically at a predetermined time or on a recurring basis. Additionally, packets may be created and replayed using the built-in facilities that are included with Capsa.

The Meterpreter module of Rapid7's Metasploit, which is an assessment tool for vulnerabilities, has the capability to store packets in a ring buffer and export them in standard pcap format for subsequent analysis. It is based on the MicroOLAP Packet Sniffer Software Development Kit. 68

A packet capture programme for TCP/IP packets, SmartSniff69 shows collected data as a sequence of client-server dialogues in ASCII mode (for text-based protocols, such as HTTP, SMTP, POP3, and FTP) or as a Hex dump, depending on the kind of protocol being examined (for non-text-based protocols). Raw sockets, the WinPcap capture driver, and the Microsoft Network Monitor driver are the three ways that SmartSniff offers for capturing packets. On earlier operating systems, the Microsoft Network Monitor driver is used.

The packet-based analytics offered by Omnipeek70 are shown in user-friendly graphical representations and are organised according to flows, which are pairs of conversations. It is able to decode more than a thousand different protocols and offers extensive packet analysis.

Moloch71 is a standalone piece of open source software that does indexed full packet capture. It uses the common pcap format for both storing packets and exporting them. Moloch has a robust graphical user interface (GUI) that is based on the web. This GUI can present information about sessions and session profiles in a tabular style, and it can graph the top unique values of fields and network connections.

PcapDB72 is a distributed complete packet capture system that optimises searches for speed and accuracy. It reorganises the packets that have been collected as they are being captured by flow, indexes the packets according to flow, and enables flow-based searches. In most cases, the indices for the collected packets take up less than one percent of the total space used by the data that was captured.

The application known as Stenographer73 is a complete packet capture programme that can quickly write captured packets to a disc. It does this by providing specific ways to get only those specific packets that are necessary for a certain study, selecting retrieving less than one percent of the packet data that is kept on disc as a result.

Packet Capture74 is a software for Android that has the power of decrypting SSL data. It is able to show the contents of the packet in either ASCII or Hex.

Tools such as CloudShark75 and pcapr are available for use by networking teams that wish to share network packets in order to conduct collaborative packet analysis.

5.2. 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 Builder77 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.

Hping78 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.

Scapy79 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.

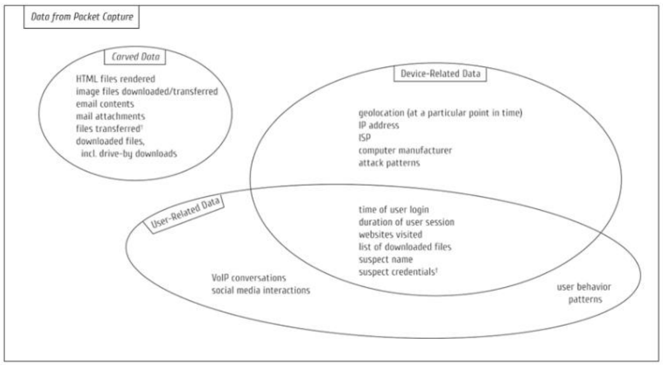


Fig. 5. Primary evidence types derived from network packets for forensic investigations.

5.3. 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.

5.3.1. 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 TruView83 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 compli- ance and security violations. Its ClearSight Analyzer can play back FTP traffic, messaging, email correspondence, and voice and video calls to quickly extract digital evidence. NetworkMiner84 acts as a passive network sniffer, which can detect operating systems, ses- sions, 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 configura- tion files to/from a PLC in a network packet capture, and extract them for analysis.

5.3.2. Tracing capabilities

When it comes to forensic applications, the amount of context that can be gained from a packet capture is determined by the logical grouping of data that is associated to packets. Many packet analyzers specialise in one or more of the following categories of network information: packet data, packet metadata, flow, session, client-server communication, and payload. Packet analyzers can track one or more levels of network information, such as packet data, packet metadata, flow, and client-server communication (Lovanshi and Bansal, 2019). Although many different types of analyzers are capable of extracting relevant information from packet capture files, certain ones, such as Tranalyzer and TCPflow, are more adept at flow analysis, whereas others, such as Fragroute and NetScout, are better suited for analysing user sessions. This is illustrated in Figure 6.
CoralReef is the greatest option to use when identifying autonomous systems, whereas ngrep is the best option to use when matching patterns or regular expressions in the data payload of packets.

6. Research challenges and future directions in packet analysis

The application of machine learning in packet analysis is developing into a sophisticated area of research that aims to solve problems such as 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 many more. In point of fact, approaches that are based on machine learning have the potential to address some of the challenges that are associated with packet analysis in relation to big network data (Yoon and DeBiase, 2018). These challenges are affecting an increasing number of packet sniffing implementations.

The examination of data packets transmitted through networks connected to the Internet of Things (IoT) is playing an increasingly significant part in the fight against cybercrime and mass surveillance. For instance, Internet of Things packet analysis may be used to assist in the detection of distributed denial-of-service (DDoS) attacks and the process of botnet formation (Salim et al., 2019). (Kumar and Lim, 2020).

Instead of using files that include packet captures from individual network segments, there is a rising demand for doing packet capturing and analysis in cloud settings due to the expanding number of cloud-based service providers. Cloud storage and cloud computing services are utilised by a variety of industries, including the government and the financial sector, cyber defence and security applications, cloud-managed services, VoIP services, and others. These services involve additional complexities on top of the source and destination IPs, protocols, and port numbers. For instance, this is exactly why Amazon introduced virtual private cloud (VPC) traffic mirroring, which makes it possible to capture and monitor AWS network traffic at scale. This is accomplished by selecting the network interface of a network resource (such as an EC2) and either an elastic network interface or load balancer on another EC2 instance. The traffic that is being delivered to the destination of the mirror is encapsulated using a VXLAN interface on the destination of the mirror. This is the case, for instance, if the network resource in a configuration is an EC2 resource, and the EC2 mirror target runs tcpdump.

Achieving a suitable balance between privacy and packet analysis has been a difficulty for a very long time (Yurcik et al., 2008), and this drives research efforts in the domain of privacy-preserving deep packet inspection. [Citation needed] (Li et al., 2017). Continued growth of the legal challenges and concerns about violating privacy with packet analysis of wireless network traffic (Ohm, 2014) and IoT devices (Vukojevi'c, 2015) calls for additional study into the topic. The accompanying technologies have to be in accordance with the ever-increasing number of rules and interception laws enacted at the national and international levels.

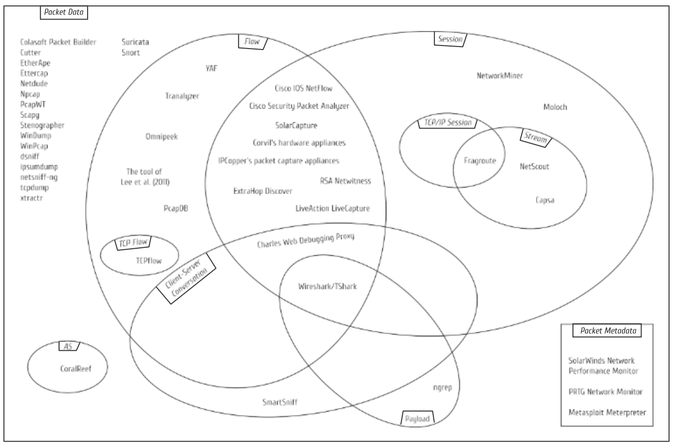


Fig. 6. Packet analyzers by the suitability for analyzing at a particular level of network information context.

7. Conclusions

In the field of network forensics, the examination of network packets is an essential step in the process of gathering the data required to achieve a comprehensive comprehension of the activities of online users that took place at a specific instant in time and to provide evidence that can be presented in court. Even though some people might be sceptical about the trustworthiness of the information that can be 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 an online activity and who was involved in it. Because using packet analysis in network forensics is different from using packet analysis in other application areas, such as intrusion detection, both the potential and limitations of packets in providing forensic evidence have been discussed. The potential of packets has been explained, but the limitations have been highlighted. There are hardware and software implementations available for packet sniffing; however, the capabilities of these tools differ substantially in terms of characteristics, supported protocols, interface, and licencing. Packet sniffing may be performed using either of these implementations. This study gave comparisons of state-of-the-art packet analyzers from a variety of perspectives, including both pros and cons.

The reader is able to acquire a solid understanding of the processes in packet analysis as well as the tools designed for packet analysis thanks to the exhaustive review that is presented in this paper. In addition, the reader is able to gain an understanding of the specific requirements of network forensics. This may be put to use in the development of original algorithms, as well as cutting-edge tools and procedures, for the purpose of packet analysis in forensic applications.

II. REFERENCES

[1]. 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), 78e87. https://doi.org/10.1145/1941487.1941508.

[2]. Agrawal, N., Tapaswi, S., 2017. The performance analysis of honeypot based intru- sion detection system for wireless network. Int. J. Wirel. Inf. Netw. 24 (1), 14e26. https://doi.org/10.1007/s10776-016-0330-3.

[3]. Al-Duwairi, B., Govindarasu, M., 2006. Novel hybrid schemes employing packet marking and logging for IP traceback. IEEE T. Parall. Distr. 17 (5), 403e418. https://doi.org/10.1109/TPDS.2006.63.

[4]. 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. 93e106. https://doi.org/10.1007/978-3-319-73951-9_5.

[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), 77e92. https://doi.org/10.1016/j.jksuci.2014.03.013.

[6]. 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. 245e282. https://doi.org/10.1007/978-3-319-72119-4_11.

[7]. Ansari, S., Rajeev, S.G., Chandrashekar, H.S., 2003. Packet sniffing: a brief intro- duction. IEEE Potentials 21 (5), 17e19. https://doi.org/10.1109/MP.2002.1166620.

[8]. Bellovin, S.M., Leech, M., 2000. ICMP traceback messages. https://www.ietf.org/ proceedings/51/I-D/draft-ietf-itrace-00.txt.

[9]. 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. 34e41. http://ceur-ws.org/ Vol-1523/STIDS_2015_T05_BenAsher_etal.pdf.

[10]. Beverly, R., Garfinkel, S., Cardwell, G., 2011. Forensic carving of network packets and associated data structures. Digit. Invest. 8, S78eS89. https://doi.org/10.1016/ j.diin.2011.05.010.

[11]. Bhandari, A., Gautam, S., Koirala, T.K., Islam, M.R., 2017. Packet sniffing and network traffic analysis using TCPda new approach. In: Kalam, A., Das, S., Sharma, K. (Eds.), Advances in Electronics, Communication and Computing. Springer, Singapore, pp. 273e280. https://doi.org/10.1007/978-981-10-4765-7_28.

[12]. Boukhtouta, A., Mokhov, S.A., Lakhdari, N.-E., Debbabi, M., Paquet, J., 2016. Network

malware classification comparison using DPI and flow packet headers.

[13]. J. Comput. Virol. Hacking Tech. 12 (2), 69e100. https://doi.org/10.1007/s11416- 015-0247-x.

[14]. Broadway, J., Turnbull, B., Slay, J., 2008. Improving the analysis of lawfully inter- cepted 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. 1361e1368. https://doi.org/10.1109/ARES.2008.122.

[15]. 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. 319e328. https://www.usenix.org/legacy/ publications/library/proceedings/lisa2000/full _papers/burch/burch_html/.

[16]. 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.

[17]. Carvalho, D.A., Pereira, M., Freire, M.M., 2009. Towards the detection of encrypted BitTorrent traffic through deep packet inspection. In: S´le¸ zak, D., Kim, T.-H.,

[18]. Fang, W.-C., Arnett, K.P. (Eds.), Security Technology. Springer, Heidelberg, pp. 265e272. https://doi.org/10.1007/978-3-642-10847-1_33.

[19]. 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.

[20]. Clarke, N., Li, F., Furnell, S., 2017. A novel privacy preserving user identification approach for network traffic. Comput. Secur. 70, 335e350. https://doi.org/ 10.1016/j.cose.2017.06.012.

[21]. 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. 51e64. https://doi.org/10.1007/978-981-13-5913-2_4.

[22]. 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.

[23]. Dong, S., Jain, R., 2019. Flow online identification method for the encrypted Skype. J. Netw. Comput. Appl. 132, 75e85. https://doi.org/10.1016/j.jnca.2019.01.007.

[24]. Duncan, R., Jungck, P., 2009. packetC language for high performance packet pro- cessing. 11th IEEE International Conference on High Performance Computing and Communications. IEEE Computer Society, Los Alamitos, CA, USA, pp. 450e457. https://doi.org/10.1109/HPCC.2009.89.

[25]. 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.

[26]. Gong, C., Sarac, K., 2005. IP traceback based on packet marking and logging. IEEE International Conference on Communications. IEEE, Piscataway, NJ, USA, pp. 1043e1047. https://doi.org/10.1109/ICC.2005.1494507.

[27]. 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. 77e81. https://doi.org/ 10.1109/CICN.2017.8319360.

[28]. Guo, Y., Gao, Y., Wang, Y., Qin, M., Pu, Y., Wang, Z., Liu, D., Chen, X., Gao, T., Lv, T.,

[29]. Fu, Z., 2017. DPI & DFI: a malicious behavior detection method combining deep packet inspection and deep flow inspection. Procedia Engineer. 174, 1309e1314. https://doi.org/10.1016/j.proeng.2017.01.276.

[30]. 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 Interna- tional Conference on Computational Intelligence and Communication Net- works. IEEE Computer Society, Los Alamitos, CA, USA, pp. 441e444. https:// doi.org/10.1109/CICN.2012.207.

[31]. 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 Infor- mation Engineering, 2. Springer, Heidelberg, pp. 135e140. https://doi.org/ 10.1007/978-3-642-30223-7_22.

[32]. Hurd, D., 2018. Endace fusion partners: redefining cybersecurity with Cisco. https:// youtu.be/iRagH8y0GBA.

[33]. 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), 8609e8619. https://doi.org/10.1007/s00500-019-03975-8.

[34]. 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. 907e914. https://doi.org/10.1007/978-981-10-7901-6_97.

[35]. 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. 1093e1102. https://doi.org/10.1007/978-3-319-24584-3_93.

[36]. Johansen, G., 2017. Acquiring host-based evidence. In: Digital Forensics and Incident Response: an Intelligent Way to Respond to Attacks. Packt Publishing, Bir- mingham, UK.

[37]. Joshi, R., Pilli, E.S., 2016. Network forensic tools. In: Fundamentals of Network Fo-rensics. Springer, London, pp. 71-93.

[38]. Jungck, P., Duncan, R., Mulcahy, D., 2011. packetC Programming. Apress. https:// doi.org/10.1007/978-1-4302-4159-1.

[39]. 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. 124e128. https:// doi.org/10.1007/978-3-642-15766-0_18.

[40]. 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. 388e393. https://doi.org/10.1109/ISPAW.2011.27.

[41]. Kim, H., Kim, E., Kang, S., Kim, H.K., 2015. Network forensic evidence generation and verification scheme (NFEGVS). Telecommun. Syst. 60 (2), 261e273. https:// doi.org/10.1007/s11235-015-0028-3.

[42]. 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, 91e102. https://doi.org/10.1016/j.comnet.2014.12.007.

[43]. Kumar, A., Lim, T.J., 2020. Early detection of Mirai-like IoT bots in large-scale net- works through sub-sampled packet traffic analysis. In: Arai, K., Bhatia, R. (Eds.), Advances in Information and Communication. Springer, Cham, pp. 847e867. https://doi.org/10.1007/978-3-030-12385-7_58.

[44]. 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. 51e63. 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 Commu- nication, Networking, and Digital Contents. Springer, Heidelberg, pp. 33e40.

[45]. https://doi.org/10.1007/978-3-642-35594-3_5.

[46]. Li, J., Su, J., Wang, X., Sun, H., Chen, S., 2017. CloudDPI: cloud-based privacy-pre- serving deep packet inspection via reversible sketch. In: Wen, S., Wu, W., Castiglione, A. (Eds.), Cyberspace Safety and Security. Springer, Cham, pp. 119e134. https://doi.org/10.1007/978-3-319-69471-9_9.

[47]. 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.

[48]. 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. 195e204. https://doi.org/10.1007/978- 981-13-6351-1_15.

[49]. 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. 385e392. https://doi.org/10.1007/978-3-642-24037-9_38.

[50]. Mielczarek, W., Mon´, T., 2015. USB data capture and analysis in Windows using USBPcap and Wireshark. In: Gaj, P., Kwiecien´, A., Stera, P. (Eds.), Computer Networks. Springer, Cham, pp. 431e443. https://doi.org/10.1007/978-3-319- 19419-6_41.

[51]. Murugesan, V., Selvaraj, M.S., Yang, M.-H., 2018. HPSIPT: a high-precision single- packet IP traceback scheme. Comput. Network. 143, 275e288. https://doi.org/ 10.1016/j.comnet.2018.07.013.

[52]. Ndatinya, V., Xiao, Z., Manepalli, V.R., Meng, K., Xiao, Y., 2015. Network forensics analysis using Wireshark. Int. J. Secur. Netw. 10 (2), 91e106. https://doi.org/ 10.1504/IJSN.2015.070421.

[53]. Nikkel, B.J., 2005. Generalizing sources of live network evidence. Digit. Invest. 2 (3), 193e200. https://doi.org/10.1016/j.diin.2005.08.001.

[54]. Ning, J., Pelechrinis, K., Krishnamurthy, S.V., Govindan, R., 2013. On the trade-offs between collecting packet level forensic evidence and data delivery perfor- mance in wireless networks. In: Kim, D.-I., Mueller, P. (Eds.), 2013 IEEE Inter- national Conference on Communications. IEEE, Piscataway, NJ, USA, pp. 1688e1693. https://doi.org/10.1109/ICC.2013.6654760.

[55]. Ohm, P., 2014. Should sniffing Wi-Fi be illegal? IEEE Secur. Priv. 12 (1), 73e76. https://doi.org/10.1109/MSP.2014.14.

[56]. 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/.

[57]. 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, 32e46. https://doi.org/10.1016/j.jnca.2019.02.022.

[58]. Parvat, T.J., Chandra, P., 2015. A novel approach to deep packet inspection for intrusion detection. Procedia Comput. Sci. 45, 506e513. https://doi.org/10.1016/ j.procs.2015.03.091.

[59]. 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 Engi- neering and Technology, II. IEEE Computer Society, Los Alamitos, CA, USA, pp. 108e113. https://doi.org/10.1109/ICCET.2009.37.

[60]. 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. 215e229. https://doi.org/10.1145/ 2987443.2987474.

[61]. 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), 1082e1110. https:// doi.org/10.3390/app7101082.

[62]. 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.

[63]. 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.

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

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

[66]. Senthivel, S., Ahmed, I., Roussev, V., 2017. SCADA network forensics of the PCCC protocol. Digit. Invest. 22, S57eS65. https://doi.org/10.1016/j.diin.2017.06.012.

[67]. 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, 157e170. https://doi.org/10.1016/j.future.2017.10.016.

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

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

[70]. Sikos, L.F., 2019. Knowledge representation to support partially automated honey- pot analysis based on Wireshark packet capture files. In: Czarnowski, I., Howlett, R.J., Jain, L.C. (Eds.), Intelligent Decision Technologies

2019. Springer, Singapore, pp. 345e351. https://doi.org/10.1007/978-981-13-8311-3_30.

[71]. 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.

[72]. 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), 721e734. https://doi.org/10.1109/TNET.2002.804827.

[73]. 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. 878e886. https://doi.org/10.1109/INFCOM.2001.916279 .

[74]. Stalla-Bourdillon, S., Papadaki, E., Chown, T., 2014. From porn to cybersecurity passing by copyright: how mass surveillance technologies are gaining legiti- macy … the case of deep packet inspection technologies. Comput. Law Secur. Rep. 30 (6), 670e686. https://doi.org/10.1016/j.clsr.2014.09.006.

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

[76]. 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.

[77]. 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. 199e212. https://www.usenix.org/legacy/events/sec2000 /full_papers/ stone/stone.pdf.

[78]. Sy, D., Bao, L., 2006. CAPTRA: coordinated packet traceback. In: 5th International Conference on Information Processing in Sensor Networks. ACM, New York, pp. 152e159. https://doi.org/10.1145/1127777.1127803.

[79]. 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. 195e207. https://doi.org/10.1007/ 978-3-642-24212-0_15.

[80]. 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 Com- puter Society, Los Alamitos, CA, USA. https://doi.org/10.1109/HICSS.2007.617.

[81]. 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. 183e197. https://doi.org/ 10.1007/978-3-319-99277-8_11.

[82]. Vukojevi´c, 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. 1338e1343. https://doi.org/10.1109/ MIPRO.2015.7160482.

[83]. 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. 1160e1165. https://doi.org/10.1109/ICC.2014.6883478.

[84]. 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), 567e580. https://doi.org/10.1109/TPDS.2008.132.

[85]. 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 Com- puter Society, Los Alamitos, CA, USA, pp. 515e520. https://doi.org/10.1109/ WAINA.2018.00137.

[86]. 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. 245e251. https://doi.org/10.1007/978-981-13-1328-8_31.

[87]. Yin, C., Wang, H., Yin, X., Sun, R., Wang, J., 2018. Improved deep packet inspection in data stream detection. J. Supercomput. 75 (8), 4295e4308. https://doi.org/ 10.1007/s11227-018-2685-y.

[88]. 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 e BigData 2018. Springer, Cham, pp. 43e56. https://doi.org/10.1007/978-3-319-94301-5_4.

[89]. 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, 1209e1216. https://doi.org/10.1016/j.procs.2018.04.331.

Cite this article as :

Aniruddha R. Jaipurkar, Dr. Nilesh Marathe, "An Analytical Review on Packet Analysis for Network Forensics and Deep Packet Inspection in Network", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Online ISSN : 2394-4099, Print ISSN : 2395-1990, Volume 9 Issue 4, pp. 147-167, July-August 2022.
Journal URL : https://ijsrset.com/IJSRSET229434