Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 10, No.2, October : 2024

EVALUATION OF FUTURE PERSPECTIVES ON SNORT AND WIRESHARK AS TOOLS AND TECHNIQUES FOR INTRUSION DETECTION SYSTEM

Sudhanshu Sekhar Tripathy, Research Scholar, Dept.Of Computer Science and Engineering, C.V Raman Global University Bhubaneswar Odisha
Bichitrananda Behera, Assistant Professor, Dept.Of Computer Science and Engineering, C.V Raman Global University Bhubaneswar Odisha.

ABSTRACT

The increasing reliance on inter-organizational information exchange has raised significant concerns about the security of data and network infrastructures. Network monitoring plays a crucial role in mitigating these concerns, with tools like Wireshark and Snort forming the backbone of Intrusion Detection Systems (IDS). Initially developed as a packet inspection application, Wireshark is widely regarded for its user-friendly interface and intuitive packet-enhancement features, making it effective for classifying various types of network traffic. This research explores the practical application of Wireshark for network investigation, evaluating its role in conjunction with Snort to enhance IDS capabilities. The study examines potential improvements in these tools for heightened network security and their adaptability to emerging cyber threats. An experiment was conducted to assess the effectiveness of intrusion detection through real-time packet analysis, demonstrating the reliability of intrusive packet authentication within network environments. Wireshark was employed for real-time traffic inspection, capturing and analyzing packets, while Snort was used as the primary tool for detecting intrusions. The integration of Syslog and Snort facilitates the exchange of critical intrusion-related data, including packet counts, analysis of IPv4 packet conversations, and expert data on suspicious traffic. This study also focuses on the analysis of RSA-encrypted traffic and the evaluation of Local Area Networks (LAN) for signs of intrusion. Further, Wireshark's capabilities in monitoring and analyzing network activity were used to inspect TCP flags, generate I/O graphs for transmitted packet data, and produce TCP stream flow graphs for detecting intrusions. Additionally, the study includes TLS handshake analysis to identify abnormal or malicious network behavior. The use of ping requests from the attacker’s IP address to the victim’s IP address is highlighted as a method for detecting ongoing malicious activity. Through packet analysis, network traffic is classified as either malformed or well-formed, aiding in the identification of security breaches. Wireshark's in-depth packet inspection enables the detection of unauthorized access from both secure and insecure devices. This research not only explores Wireshark's utility in network intrusion detection but also evaluates emerging trends and challenges associated with IDS technologies. The findings contribute valuable insights for advancing future IDS research, particularly in adapting to the evolving landscape of network security threats. This technical evaluation highlights the importance of continuous development in tools like Wireshark and Snort to keep pace with the dynamic nature of cyberattacks, ensuring robust defense mechanisms for secure data transmission and network integrity.

Keywords:
IDS, Wireshark, Snort, Packet Analysis, Protocol Hierarchy, Event Log Analyzer, Malformed Packet.

1. Introduction

Wireshark, originally launched by Gerald Combez in 1997 under the name Ethereal, was designed as a comprehensive network packet analysis tool. The first official version, 0.2.0, was released in July 1998. After a decade of ongoing development, Wireshark 1.0 was introduced in 2008, marking its first full release. This release coincided with the inaugural Sharkfest, a conference for Wireshark developers and users. In 2015, Wireshark 2.0 introduced a modernized user interface, improving usability. Wireshark version 2.2.9, released on May 11, 2023, included significant GUI updates and support for Internet Protocol Version 6 (IPv6). The most recent version before Wireshark 4.4.1 was

4.4.0, launched on August 29, 2024. Wireshark 4.4.1, released on October 9, 2024, is the latest stable version and includes advanced features surpassing prior releases. The rise in internet usage has increased security risks such as unauthorized data access, network integrity compromises, and confidentiality breaches. To mitigate these risks, organizations use firewalls, antivirus software, and Intrusion Detection Systems (IDS). Network-based IDS (NIDS) monitor network traffic for potential threats, while Host-based IDS (HIDS) safeguard individual devices. With the increasing sophistication of cyberattacks, tools like Snort, an open-source IDS, are essential for identifying and mitigating intrusions. Snort performs real-time packet analysis and uses a signature-based detection method to identify known threats. It also generates logs and alerts for suspicious activities. Complementary tools like Wireshark offer in-depth traffic analysis, making them invaluable for detecting vulnerabilities and analyzing attack patterns. As cyber threats evolve, the continuous development of IDS tools is critical to maintaining robust and scalable network defenses, ensuring organizations can proactively prevent potential breaches and secure sensitive data.

Snort-IDS uses predefined rules for data packet traffic and generates alerts when a packet matches these rules. It is effective against various types of attacks, enabling content searching and protocol analysis. Snort functions by sniffing network packets and comparing each packet with its rule set to detect malicious activity. The increased use of mobile devices has intensified the need for research in security, particularly concerning the unauthorized entry of malicious users or dangerous data packets. Data packets, the core units of communication systems, are crucial for network security as they facilitate data flow between devices, including the hardware address and protocols. Packet sniffing identifies suspicious packets by examining their contents, focusing on the data segment, logging the information, and analyzing the data as depicted in Figure 1. A packet analyzer intercepts and decodes data from a network, ensuring its integrity and compliance with security protocols.

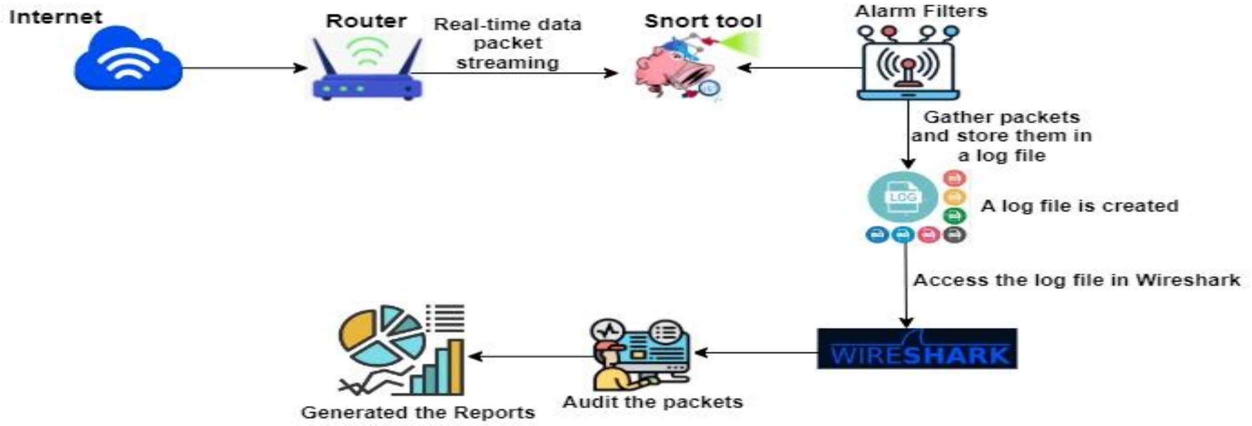


Figure 1: The design of the Snort tool and the Wireshark-based evaluation of the network process

Figure 1 appears to illustrate the working process of a network monitoring and intrusion detection system utilizing tools like Snort and Wireshark. Here's an outline of the key concepts presented:

- **Internet to Router**: The data flow begins from the internet, where network traffic is streamed in real-time via the router. The router serves as the first checkpoint where data packets pass through.
- **Snort Tool**: After data is routed, the Snort tool captures and analyzes this traffic. Snort is a network intrusion detection system (NIDS) that monitors traffic for suspicious patterns and potential threats.
- **Alarm Filters**: Snort then filters alarms based on predefined rules or patterns. This step is critical for identifying potentially malicious activities from legitimate traffic.
- **Packet Logging:** Once alarms are triggered, relevant packets are gathered and stored in a log file for further inspection. This log file keeps a record of all suspicious or anomalous activity.

- **Wireshark Access**: The log file created by Snort is then accessed using Wireshark, which is a tool for deep packet analysis. Wireshark helps to examine the contents of network packets in more detail, aiding in the forensic investigation.
- **Audit and Reporting**: After packets are examined, the auditing phase begins, where security experts review the findings. Reports are then generated based on the analysis, providing insights into the network's security and potential vulnerabilities.

This process outlines the systematic approach to monitoring, capturing, analyzing, and reporting network traffic for intrusion detection and cybersecurity.

1.1 Intro. for Intrusion Detection System

An Intrusion Detection System (IDS) is a tool designed to monitor network and system traffic for suspicious activity or security breaches. It analyzes data to detect unauthorized access, abnormal behavior, or deviations from expected patterns. When a threat is identified, the IDS generates logs and alerts for administrators to investigate, as depicted in Figure 2. There are two primary types: signature-based IDS, which compares traffic against a database of known threat patterns, and anomaly-based IDS, which identifies deviations from normal behavior to detect unknown threats. While signature-based IDS is effective for known attacks, anomaly-based IDS can detect new threats but may produce more false positives [1].

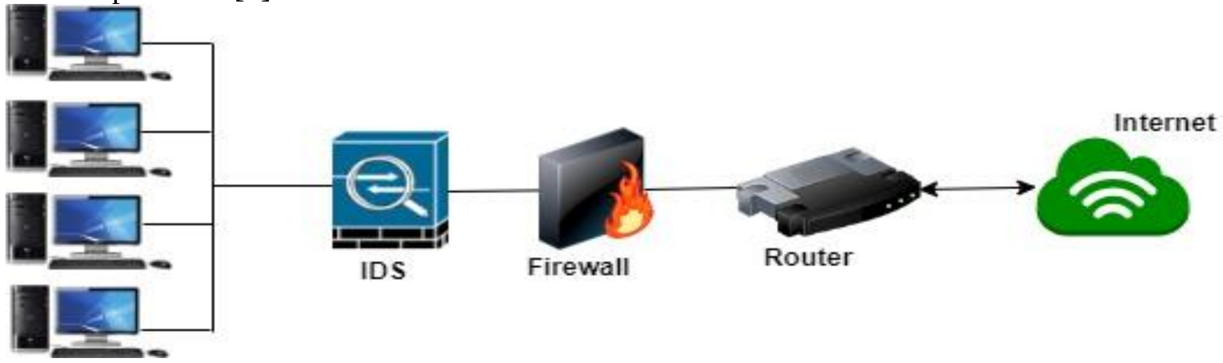


Figure 2: Intrusion Detection System

Local Computers (Hosts)

Figure 2, depicts a network architecture focused on intrusion detection and security, showing the interaction between several components of a secure network. Here's an outline of the key concepts presented.

- **Local Computers (Hosts):** Multiple computers within a local network are connected to the infrastructure. These endpoints represent the internal devices that send and receive data through the network.
- **Intrusion Detection System (IDS):** The traffic from the computers is monitored by the IDS, which scans for malicious activity or violations of security policies. The IDS is responsible for identifying possible security threats by analyzing data packets as they flow through the network.
- **Firewall**: After the IDS, traffic passes through a firewall, which acts as a barrier between the internal network and external networks. The firewall enforces security rules by blocking or permitting network traffic based on predefined policies, providing a layer of defense against unauthorized access.
- **Router**: The router directs traffic between the internal network and the external internet. It handles the routing of data packets, ensuring that information from internal computers is sent to its appropriate destination on the internet, and vice versa.
- **Internet Connection**: The internet represents the external network, where data exchanges take place beyond the organization's boundaries. It is the entry point for potential threats, which the IDS and firewall work to mitigate.

This network layout shows the sequence of security mechanisms—from intrusion detection to firewall enforcement—that protect a local network as data flows between internal computers and the broader internet. It highlights how an IDS works in tandem with a firewall to monitor and control traffic, ensuring that only legitimate data passes through the network.

1.2 Types of IDS

Network Intrusion Detection System (NIDS): A NIDS is designed to monitor and analyze network traffic for signs of malicious activity. It evaluates both connection-oriented (TCP) and connectionless (UDP) traffic to detect unauthorized access or suspicious behavior across the network. By examining packets flowing through the network in real-time, NIDS can identify potential threats and alert administrators to any anomalies or security breaches. This proactive monitoring helps organizations respond quickly to emerging threats and maintain the integrity of their network environments.

Host-Based Intrusion Detection System (HIDS): A HIDS operates by monitoring specific host machines. It tracks activities such as file changes, system logs, and other key indicators of potential breaches on individual devices. This system provides protection by focusing on anomalies and unauthorized actions occurring on a single host.

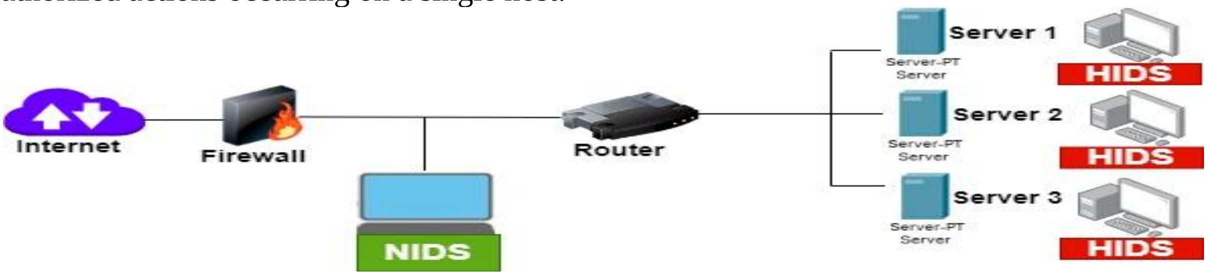


Figure 3: A general Overview of NIDS and HIDS

Figure 3 represents a network security architecture that combines both Network Intrusion Detection Systems (NIDS) and Host-Based Intrusion Detection Systems (HIDS) to secure different layers of the network. Here's an outline of the key concepts presented.

➢ **Internet to Firewall**: Data from the internet enters the network, passing first through the firewall, which is configured to block or allow traffic based on predefined security rules. The firewall acts as the first line of defense against unauthorized access.

➢ **NIDS (Network Intrusion Detection System):** Positioned after the firewall, the NIDS monitors all incoming and outgoing network traffic. It scans packets for suspicious behavior, allowing early detection of potential threats before they reach internal servers. NIDS primarily focuses on detecting anomalies across the network as a whole, making it suitable for broader network-level monitoring.

➢ **Router**: After the traffic is filtered by the firewall and analyzed by the NIDS, it reaches the router. The router is responsible for directing network traffic to the correct destination within the local network.

➢ **Servers with HIDS (Host-Based Intrusion Detection System):**

- Each server in the internal network (Server 1, Server 2, Server 3) is equipped with a HIDS.
- HIDS works at the host level, monitoring activities like file system modifications, process behavior, and logs for any signs of malicious activity. HIDS provides more granular control by focusing on potential intrusions specific to the host it is installed on.
- This host-based protection layer complements the broader network-level protection offered by the NIDS.

2. Literature survey

In [2], the author presented the properties of Snort, a Network Intrusion Detection System (NIDS) that detects intrusions using predefined rules and notifies users through alert messages. Snort generates log files that can be exported to Wireshark for further analysis. Wireshark provides detailed information about network packets, Ethernet frames, and protocols, and includes an I/O graph that summarizes packet flow and provides expert insights into network activity.

In [3], the author demonstrated Wireshark's effectiveness as a network sniffing tool, particularly in detecting malicious packets. Through experimentation on a live network, Wireshark was identified as a strong candidate for further development into a reliable intrusion detection system. The paper emphasizes Wireshark's flexibility as an open-source tool, enabling developers to integrate intrusion detection functionalities. Additionally, the authors noted that Wireshark is a powerful instrument for handling and interpreting packet data, primarily utilized for Access Control List (ACL) filtering in this study. Its various filtering options, including substring, packet size, and protocol filtering, further enhance its potential as an intrusion detection tool.

In [4], the author implements live network traffic recording, utilizing Wireshark and Snort for in-depth packet analysis. While Wireshark provides detailed visibility into network traffic, it does not trigger alarms or security measures against unauthorized access. In contrast, Snort actively prevents intrusions and generates alerts for unusual activity. The analysis includes file graphs that offer insights into network dynamics and potential issues, highlighting the complementary roles of both tools in enhancing network security.

In [5], the author implemented Wireshark for network protocol analysis and identified various attacks, including port inspection, ICMP attacks, hidden FTP channels, and BitTorrent services. The study highlighted how Wireshark's packet analysis uncovers multiple security threats present in networked computer systems.

In [6], the author proposed a technique to evaluate the effectiveness and precision of distributed denial of service (DDoS) attacks through TCP flooding. The study examines key factors that influence Snort's detection capabilities and suggests improvements to enhance the Intrusion Detection System (IDS). It demonstrates that upgraded tools can significantly improve Snort's efficiency in handling such attacks.

In [7], the authors conducted a comprehensive network protocol analysis to gather data for technical packet analysis, focusing on security, network sniffing, and protocol analysis to identify network attacks.

In [8], the author explores the significance of intrusion detection systems (IDS) in today's corporate environment, focusing on their lifecycles, domains, attack types, and various tool types. The article emphasizes the necessity for network users to adopt security precautions and outlines the distinct phases in the IDS lifecycle. Additionally, it discusses the application of selective feedback methods in data mining algorithms for recognition and classification systems, aiming to enhance classification accuracy continuously.

This research project [9] simulates a ping flood scenario by executing the ping command on the operating system while simultaneously running Wireshark on the target system. Analyzing acknowledged ping packets over time aids in detecting flooding attacks; however, Wireshark's one-port reception can lead to inaccuracies in handling requests. The paper discusses Wireshark's functionality, drawbacks, and potential improvements. It emphasizes Wireshark's utility for network administrators, particularly in tracing back attacks, as attackers often use zombie machines in denial-of-service attacks. Wireshark employs ICMP to trace packet paths, generates alerts when multiple packets arrive from the same source, and uses a packet marking algorithm based on unique identifiers and receipt times, functioning effectively as an intrusion detection system.

In [10], the signature-based approach is highlighted as an effective method for identifying threats in intrusion detection systems (IDS). The free SNORT program can implement this technique, utilizing rules to detect various types of attacks. The study tested alerting mechanisms using the MIT-DARPA 1999 dataset, which comprises 1,252,412 packets, focusing on DoS attacks and network scanning. The IDS achieved an accuracy of 98.10% with a true positive rate of 100%.

This work [11] aims to enhance Snort IDS efficiency by designing and implementing a real-time intrusion detection system. The system utilizes pre-built and customizable rules to prevent potential attacks on network systems. The article discusses installation, components, features, and products that integrate with Snort. The author tested Snort alert interpretation on Kali Linux, intending to incorporate the proposed design into Snort for improved detection efficiency and reduced false alerts.

This work [12] focuses on network monitoring, identifying network attack types, and utilizing software tools to protect communication networks. Network security is a top priority for companies, which employ firewalls, VPNs, and encryption techniques. Despite these measures, hackers can still breach networks, making network monitoring tools like Wireshark and Snort essential for detecting potential intrusions through graphical analysis of network activity. The network infrastructure is improving security and performance through monitoring with Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS). Wireshark and Snort are effective tools for intrusion detection, offering features such as color rules, expert information, network monitoring, and firewall Access Control List (ACL) rules. Snort also allows for security policy modifications based on specific requirements.

The authors [13] discuss the use of real-time and offline data mining in Intrusion Detection Systems (IDS), emphasizing the importance of offline data analysis for successful detection tasks. They explore detection rule analysis to transfer logs from remote sites to a centralized system for evaluation. Additionally, they propose innovative methods for network anomaly detection using packet header values in real-time data mining analysis within IDS.

Sourcefire developed Snort, an open-source network intrusion prevention and detection system (IPS/IDS) that combines signature, protocol, and anomaly-based inspection. It is the most widely used IPS/IDS technology globally, primarily relying on these detection methods. Snort compares network traffic signatures with predefined entries in its library, ensuring a secure and efficient network [14]. However, Snort has limitations; while signature-based detection enhances performance by matching patterns, it cannot identify new attack types that are not included in its database.

Intrusions threaten the availability, confidentiality, and integrity of computing or networking resources. An Intrusion Detection System (IDS) detects and reports these intrusions to an administrator, while an Intrusion Prevention System (IPS) not only detects but also prevents intrusions. Both IDS and IPS are designed to address gaps left by firewalls [15], with IDS primarily focused on identifying threats within a network.

According to the authors [16], network process monitoring is increasingly recognized as a crucial tool for enhancing cyber infrastructure security. It utilizes stateful and pattern matching to detect intruders, and IDS techniques should be affordable, workable, and commercially viable. A combination of IDS and IPS provides layered security, reducing risks if the IPS fails to stop an intrusion. IDS technology offers visibility and benefits related to network monitoring, enabling informed decision-making and the creation of security policies based on measurable data, as noted by the author [17]. Effective security combines visibility and control, allowing for data storage for later analysis and real-time monitoring of network activity.

Traditional intrusion detection methods have limitations; however, data mining and network behavior analysis (NBA) can enhance the effectiveness of detecting intrusions, according to the author [18]. Two types of intrusion detection systems (IDS) are misuse detection systems and anomaly detection systems. Misuse detection involves maintaining a database of known intrusion signatures, alerting security analysts when user behavior matches these stored patterns, which informs their response based on the attack's nature. In contrast, anomaly detection identifies unusual intrusions by creating profiles of typical network behavior and flagging patterns that significantly deviate from these profiles.

Wireshark is a widely used network protocol analyzer that provides detailed insights into network activity. Although it is classified as an intrusion detection tool, it is not an actual intrusion detection or prevention system. The expert information noted by the authors [19] serves as a log of the anomalies Wireshark identifies in captured network data.

Expert reports in Wireshark include conversations (grey), remarks (cyan), alerts (yellow), and errors (red), as shown in Figure 20. Intrusion detection is possible in the conversation section by analyzing SYN, SYN+ACK, and ACK messages in TCP connections. To identify the origin IP of a DoS attack, examine the chat section and utilize Firewall ACL rules. The flow graph section illustrates communication between distinct IPs. If a TCP flow graph displays only SYN messages, this indicates a DoS attack. Identifying intrusions also involves analyzing the conversation section in Figure 29, which shows traffic between specific endpoints, capturing all communication between two IP addresses.

Network monitoring is the most effective way to prevent intrusions. The authors [20] cover various software tools useful for monitoring network activities, including RRDTOOL, Kiwi Syslog, Splunk, Nagios, RANCID, SNORT, NFDUMP/NFSEN, SmokePing, Munin, NetDisco, WhatsUp Gold, ZABBIX, NAV, NetXMS, ZENOSS, AirWave, Cisco WCS, 7Signal Sapphire, Big Sister, Cacti, Cricket, and MRTG. Additionally, open-source intrusion detection tools like SURICATA, Bro, KISMET, OSSEC, Samhain, and Open DLP are also effective for enhancing network security.

3. IDS Architecture

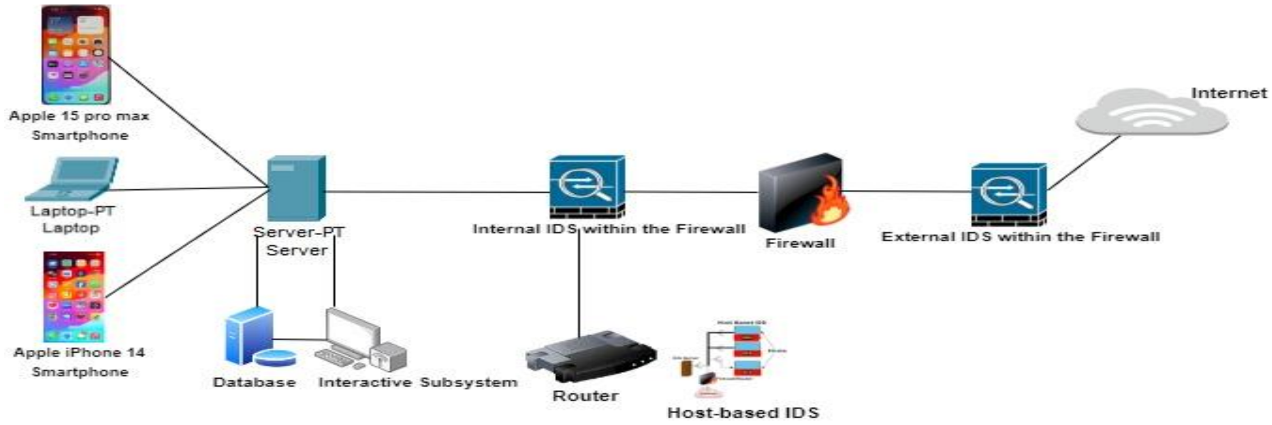


Figure 4: Architecture of an IDS

Figure 4 illustrates a network security architecture featuring various components. Two smartphones (Apple iPhone 15 Pro Max and iPhone 14) and a laptop connect to a server, which is linked to a database and interactive subsystem. The server is connected to internal and external intrusion detection systems (IDS), separated by a firewall. A host-based IDS monitors specific system activities. A router provides a link to the external network. The system is protected by both internal and external IDS to monitor traffic within and outside the firewall, ensuring security.

4. Intrusion Detection Utilizing Network Monitoring Tools for Enhanced Security

In this research, we utilize Snort and Wireshark to detect intrusions. A brief overview of each tool is provided below, as shown in Figure 8. Wireshark is a widely used open-source network packet analyzer that captures and examines data packets, making it valuable for network troubleshooting, security analysis, and understanding protocol operations [20]. Snort, a powerful security application, functions as a Network Intrusion Detection System (NIDS), packet sniffer, and packet logger. Additionally, Snort supports various add-on tools for log management, rule set maintenance, and alerting, enhancing its capability as a critical component in an organization's security infrastructure [21].

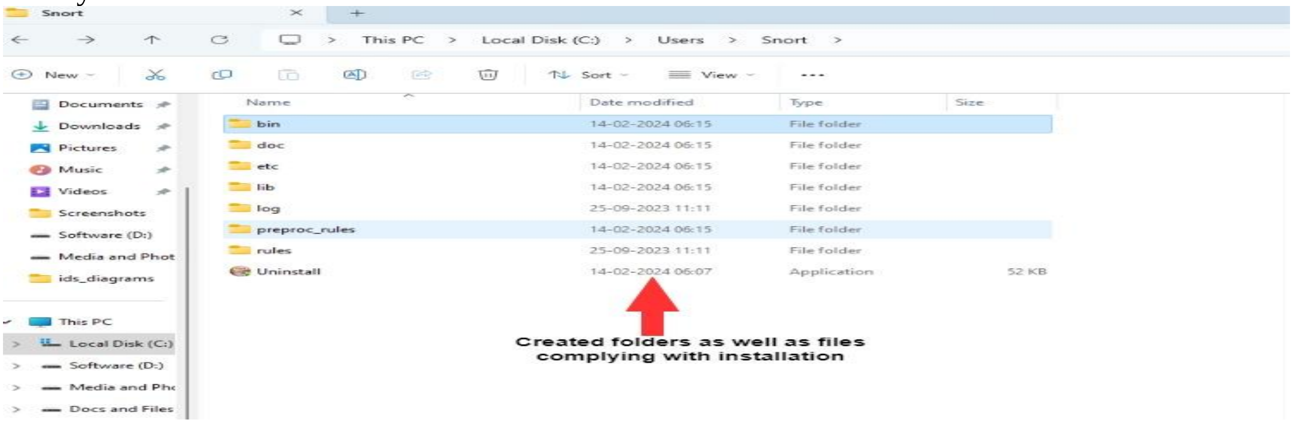
4.1 Practical Approach to Intrusion Detection Using Wireshark

Wireshark is a highly adaptable network protocol analyzer that can be configured to operate as an Intrusion Detection System (IDS). Users can customize various features, including coloring rules, to highlight common packet anomalies such as malformed packets and checksum errors, as illustrated in Figure 20.

This adaptability significantly enhances its effectiveness in detecting potential security threats and network irregularities. In addition to its analysis capabilities, Wireshark provides intrusion detection features, including the ability to permit or block packets to specific IP addresses using its integrated firewall. It also offers detailed insights into malformed packets, assessing the severity of packet errors and providing packet counts. Graph analysis in Wireshark facilitates the identification of communication patterns between IP addresses. This feature allows users to visualize conversations, displaying the volume of bytes and packets exchanged between systems. A surge in ping requests from a single IP address may signal a Denial of Service (DoS) or Distributed Denial of Service (DDoS) attack.

4.2 An Effective Approach to Intrusion Detection with Snort

To install and configure Snort on a Windows machine, download the software from the designated source and follow the provided installation instructions. The WinPcap software is necessary for packet capture interpretation. Once installation is complete, navigate to the specified folders, as shown in Figure 5. Snort is a lightweight intrusion detection tool that captures and analyzes network packets, comparing them against user-defined rules. When a match is detected, Snort generates alerts. These rules are stored in a text file associated with the snort.conf configuration file, which encompasses all Snort settings. A few command-line instructions are utilized to launch Snort and assess network activity.



Name	Date modified	Type	Size
bin	14-02-2024 06:15	File folder	
doc	14-02-2024 06:15	File folder	
etc	14-02-2024 06:15	File folder	
lib	14-02-2024 06:15	File folder	
log	25-09-2023 11:11	File folder	
preproc_rules	14-02-2024 06:15	File folder	
rules	25-09-2023 11:11	File folder	
Uninstall	14-02-2024 06:07	Application	52 KB

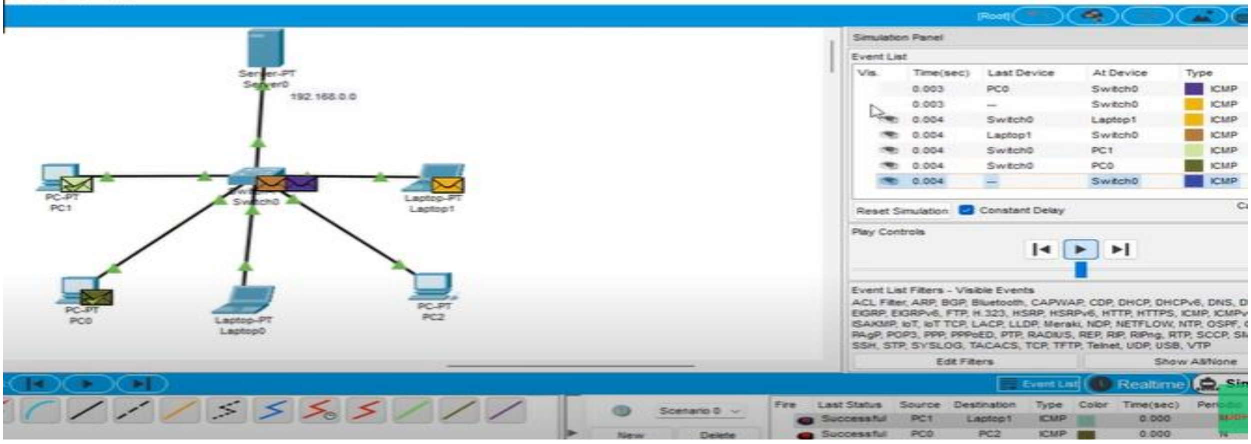
Created folders as well as files complying with installation

Figure 5: An Effective Method for Detecting Intrusions Using Snort

5. Methodology and experimental analysis

This section details the necessary hardware and software configurations for conducting experiments and evaluates the reports generated by the Wireshark tool in the context of intrusion detection. The experiments are performed on a Kali Linux environment, utilizing VMware Workstation in conjunction with a Windows 11 host system. Additionally, the draw.io application is employed to create visual representations and tactical diagrams for the evaluation procedures related to the detection mechanism. A typical Intrusion Detection System (IDS) is classified as a Network Intrusion Detection System (NIDS). NIDS operates across all layers of the OSI model, analyzing network traffic to ascertain its intended purpose and identifying anomalous behavior. These systems are designed for easy deployment throughout networks, allowing them to process and monitor traffic from multiple devices concurrently, as outlined in the functionality of the proposed system.

I. The client initiates a service request to the server, as illustrated in Figure 6-9. In response, the server processes the request and delivers the corresponding service to the client.



Event List

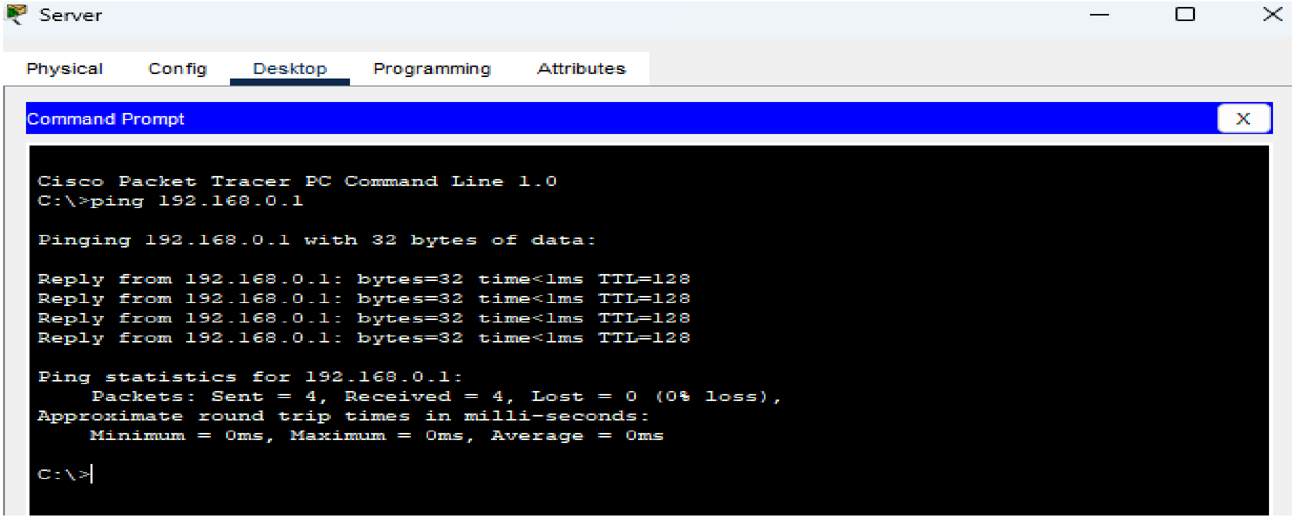
Vis.	Time(sec)	Last Device	At Device	Type
	0.003	PC0	Switch0	ICMP
	0.003	—	Switch0	ICMP
	0.004	Switch0	Laptop1	ICMP
	0.004	Laptop1	Switch0	ICMP
	0.004	Switch0	PC1	ICMP
	0.004	Switch0	PC0	ICMP
	0.004	—	Switch0	ICMP

Event List Filters - Visible Events
ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, D
EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv
ISAKMP, IoT, IoT TCP, LACP, LLDP, Meraki, NDP, NETFLOW, NTP, OSPF, O
PAgP, POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SM
SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic
●	Successful	PC1	Laptop1	ICMP		0.000	
●	Successful	PC0	PC2	ICMP		0.000	N

Figure 6: Configuration setup for packet transmission in response to a user service request from the

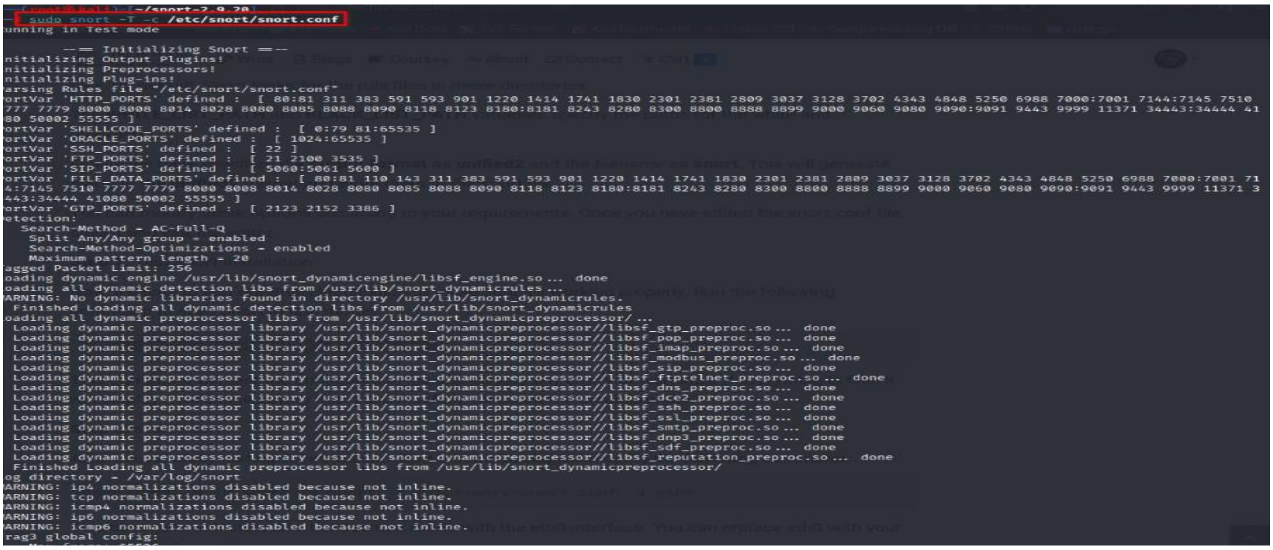
server.



```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.0.1

Pinging 192.168.0.1 with 32 bytes of data:

Reply from 192.168.0.1: bytes=32 time<1ms TTL=128
Reply from 192.168.0.1: bytes=32 time<1ms TTL=128
Reply from 192.168.0.1: bytes=32 time<1ms TTL=128
Reply from 192.168.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
```

Figure 7: Checking the ping connection on the server

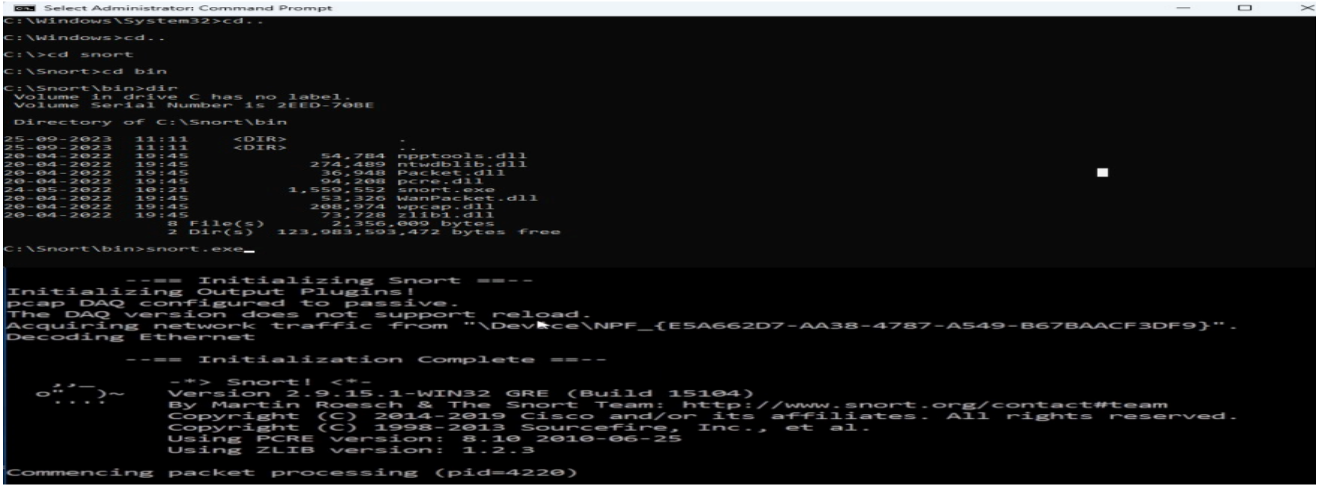
Figure 7 shows the verification of the server's connectivity through a ping test. This process involves sending Internet Control Message Protocol (ICMP) echo request packets from the client to the server and measuring the round-trip time for responses. A successful ping indicates that the server is reachable and operational, providing an essential diagnostic tool for assessing network connectivity and performance.



```
─(root@kali)-[~/snort-2.9.20]
─ sudo snort -T -c /etc/snort/snort.conf
unning in Test mode

        --== Initializing Snort ==--
nitializing Output Plugins!
nitializing Preprocessors!
nitializing Plug-ins!
arsing Rules file "/etc/snort/snort.conf"
ortVar 'HTTP_PORTS' defined :  [ 80:81 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001 7144:7145 7510
777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 8243 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:34444 41
80 50002 55555 ]
ortVar 'SHELLCODE_PORTS' defined :  [ 0:79 81:65535 ]
ortVar 'ORACLE_PORTS' defined :  [ 1024:65535 ]
ortVar 'SSH_PORTS' defined :  [ 22 ]
ortVar 'FTP_PORTS' defined :  [ 21 2100 3535 ]
ortVar 'SIP_PORTS' defined :  [ 5060:5061 5600 ]
ortVar 'FILE_DATA_PORTS' defined :  [ 80:81 110 143 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001 71
4:7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 8243 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 3
443:34444 41080 50002 55555 ]
ortVar 'GTP_PORTS' defined :  [ 2123 2152 3386 ]
etection:
  Search-Method = AC-Full-Q
   Split Any/Any group = enabled
   Search-Method-Optimizations = enabled
   Maximum pattern length = 20
agged Packet Limit: 256
oading dynamic engine /usr/lib/snort_dynamicengine/libsf_engine.so ...  done
oading all dynamic detection libs from /usr/lib/snort_dynamicrules ...
ARNING: No dynamic libraries found in directory /usr/lib/snort_dynamicrules.
 Finished Loading all dynamic detection libs from /usr/lib/snort_dynamicrules
oading all dynamic preprocessor libs from /usr/lib/snort_dynamicpreprocessor/ ...
 Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocessor//libsf_gtp_preproc.so ...  done
 Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocessor//libsf_pop_preproc.so ...  done
 Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocessor//libsf_imap_preproc.so ...  done
 Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocessor//libsf_modbus_preproc.so ...  done
 Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocessor//libsf_sip_preproc.so ...  done
 Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocessor//libsf_ftptelnet_preproc.so ...  done
 Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocessor//libsf_dns_preproc.so ...  done
 Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocessor//libsf_dce2_preproc.so ...  done
 Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocessor//libsf_ssh_preproc.so ...  done
 Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocessor//libsf_ssl_preproc.so ...  done
 Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocessor//libsf_smtp_preproc.so ...  done
 Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocessor//libsf_dnp3_preproc.so ...  done
 Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocessor//libsf_sdf_preproc.so ...  done
 Loading dynamic preprocessor library /usr/lib/snort_dynamicpreprocessor//libsf_reputation_preproc.so ...  done
 Finished Loading all dynamic preprocessor libs from /usr/lib/snort_dynamicpreprocessor/
og directory = /var/log/snort
ARNING: ip4 normalizations disabled because not inline.
ARNING: tcp normalizations disabled because not inline.
ARNING: icmp4 normalizations disabled because not inline.
ARNING: ip6 normalizations disabled because not inline.
ARNING: icmp6 normalizations disabled because not inline.
rag3 global config:
```

Figure 8: The computer program used for the intrusion detection systems using the command line

Figure 8 displays the Command-line interface (CLI) application employed for the configuration and management of intrusion detection systems (IDS). This program serves as a powerful tool for network administrators and security professionals, allowing them to monitor and analyze network traffic effectively. Through the CLI, users can execute various commands and scripts to configure the IDS settings, initiate real-time traffic analysis, and receive alerts on potential intrusions. Additionally, the CLI provides options for fine-tuning detection parameters, reviewing logs, and generating reports, facilitating a comprehensive approach to network security and anomaly detection.

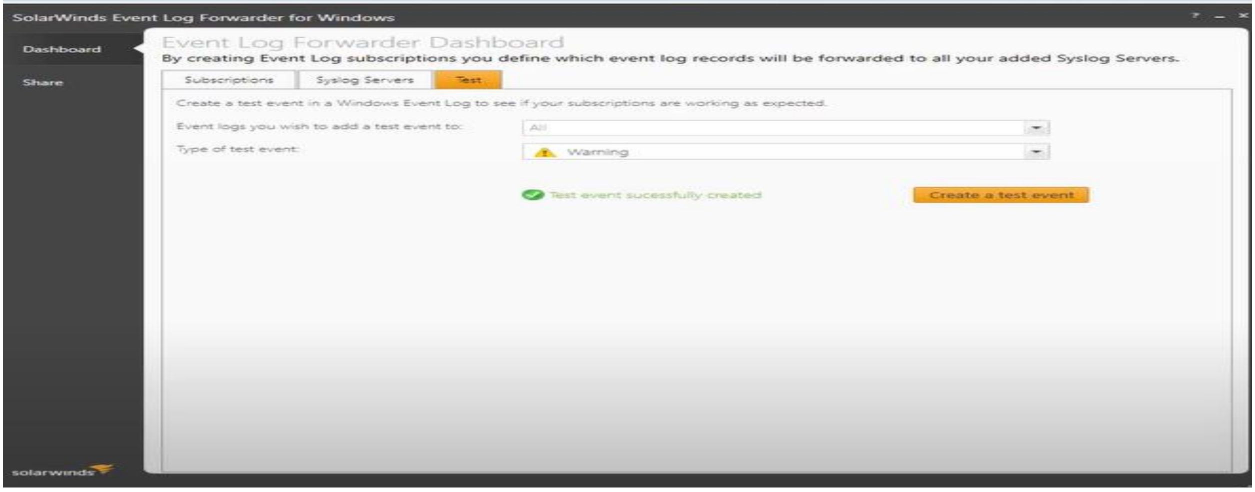


```
C:\Windows\System32>cd..

C:\Windows>cd..

C:\>cd snort

C:\Snort>cd bin

C:\Snort\bin>dir
 Volume in drive C has no label.
 Volume Serial Number is 2EED-70BE

 Directory of C:\Snort\bin

25-09-2023  11:11    <DIR>          .
25-09-2023  11:11    <DIR>          ..
20-04-2022  19:45            54,784 npptools.dll
20-04-2022  19:45           274,489 ntwdblib.dll
20-04-2022  19:45            36,948 Packet.dll
20-04-2022  19:45            94,208 pcre.dll
24-05-2022  10:21         1,559,552 snort.exe
20-04-2022  19:45            53,326 WanPacket.dll
20-04-2022  19:45           208,974 wpcap.dll
20-04-2022  19:45            73,728 zlib1.dll
               8 File(s)      2,356,009 bytes
               2 Dir(s)  123,983,593,472 bytes free

C:\Snort\bin>snort.exe_

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{E5A662D7-AA38-4787-A549-B67BAACF3DF9}".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.15.1-WIN32 GRE (Build 15104)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.3

Commencing packet processing (pid=4220)
```

Figure 9: Applying snort as a tool for intrusion

Figure 9 outlines Snort is a high-performance, open-source NIDS that inspects network packets in real-time. It uses rule-based signatures to detect anomalies, including buffer overflows, port scans, and protocol-based attacks. Snort operates in sniffing, logging, or intrusion detection modes, providing real-time alerts and comprehensive network security analysis.

II. Figure 10-11 Configuring the Event Log Forwarder Dashboard involves forwarding critical system logs for in-depth analysis. Logs are collected from endpoints, firewalls, or intrusion detection systems (like Snort), then forwarded to a centralized dashboard. By integrating Wireshark for packet capture and analysis, the system enables comprehensive monitoring of network activity, allowing for rapid identification and mitigation of potential intrusions.



By creating Event Log subscriptions you define which event log records will be forwarded to all your added Syslog Servers.

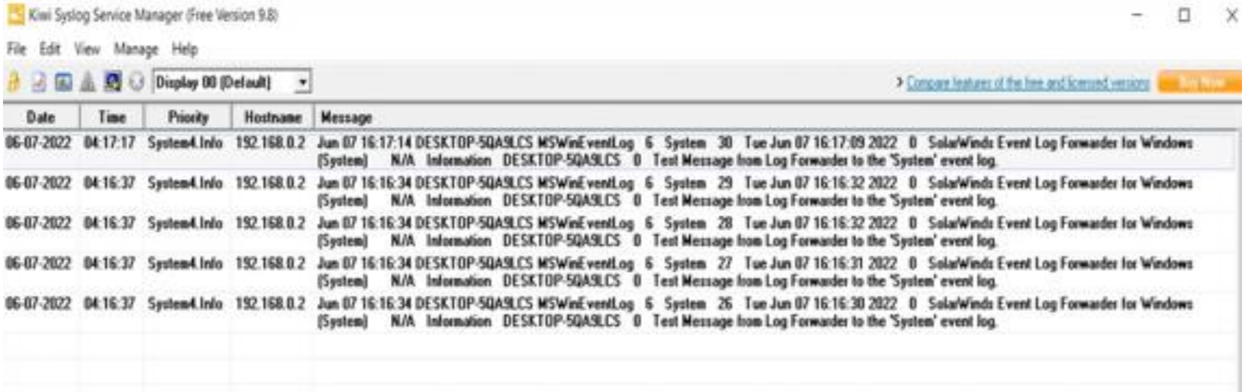
Create a test event in a Windows Event Log to see if your subscriptions are working as expected.

Event logs you wish to add a test event to: All

Type of test event: Warning

Test event sucessfully created

Figure 10: Event log Forwarder Dashboard for testing Intrusions



Kiwi Syslog Service Manager (Free Version 9.8)
File
Edit
View
Manage
Help
Display 00 (Default)
Compare features of the free and licensed versions
Buy Now

Date	Time	Priority	Hostname	Message
06-07-2022	04:17:17	System4.Info	192.168.0.2	Jun 07 16:17:14 DESKTOP-5QA9LCS MSWinEventLog 6 System 30 Tue Jun 07 16:17:09 2022 0 SolarWinds Event Log Forwarder for Windows (System) N/A Information DESKTOP-5QA9LCS 0 Test Message from Log Forwarder to the 'System' event log.
06-07-2022	04:16:37	System4.Info	192.168.0.2	Jun 07 16:16:34 DESKTOP-5QA9LCS MSWinEventLog 6 System 29 Tue Jun 07 16:16:32 2022 0 SolarWinds Event Log Forwarder for Windows (System) N/A Information DESKTOP-5QA9LCS 0 Test Message from Log Forwarder to the 'System' event log.
06-07-2022	04:16:37	System4.Info	192.168.0.2	Jun 07 16:16:34 DESKTOP-5QA9LCS MSWinEventLog 6 System 28 Tue Jun 07 16:16:32 2022 0 SolarWinds Event Log Forwarder for Windows (System) N/A Information DESKTOP-5QA9LCS 0 Test Message from Log Forwarder to the 'System' event log.
06-07-2022	04:16:37	System4.Info	192.168.0.2	Jun 07 16:16:34 DESKTOP-5QA9LCS MSWinEventLog 6 System 27 Tue Jun 07 16:16:31 2022 0 SolarWinds Event Log Forwarder for Windows (System) N/A Information DESKTOP-5QA9LCS 0 Test Message from Log Forwarder to the 'System' event log.
06-07-2022	04:16:37	System4.Info	192.168.0.2	Jun 07 16:16:34 DESKTOP-5QA9LCS MSWinEventLog 6 System 26 Tue Jun 07 16:16:30 2022 0 SolarWinds Event Log Forwarder for Windows (System) N/A Information DESKTOP-5QA9LCS 0 Test Message from Log Forwarder to the 'System' event log.

Figure 11: The Syslog Server is utilizing Snort to exchange information about intrusions

In this setup, the Syslog server collects and processes intrusion data from Snort, a network intrusion detection system. Snort monitors network traffic and identifies potential threats, sending alerts to the Syslog server as depicted in Figure 11. The server consolidates these alerts with other system logs, allowing centralized monitoring, real-time analysis, and threat correlation for better network defense.
The table 1 provides a detailed overview of the compatibility of various physical network interfaces across multiple operating systems, such as Linux, Windows, macOS, and Unix-based systems like FreeBSD, OpenBSD, and Solaris. Ethernet, a universally supported interface, is available on all platforms, reflecting its dominance in network communications. ATM (Asynchronous Transfer Mode) support is mainly seen in Linux and Solaris, crucial for high-speed data transfer in broadband applications. CiscoHDLC, used for point-to-point WAN connections, shows broad support on Linux and BSD systems, while Frame Relay, designed for cost-efficient data transmission, has limited OS support. Technologies such as FDDI (Fiber Distributed Data Interface) and Token Ring, though legacy, maintain compatibility with Linux and certain Unix systems. PPP (Point-to-Point Protocol), critical for dial-up connections, is widely supported across Linux, macOS, and Windows. WLAN interfaces are well-integrated across most platforms, while Bluetooth and IrDA (Infrared Data Association) show narrower support, mainly within Linux-based systems. VLAN tagging, essential for virtual network segmentation, is consistently supported across most OS platforms.
Wireshark is a powerful tool that can efficiently capture and analyze network traffic from various platforms and network types in Table 1:

Physical Interface	AIX	FreeBSD	HP-UX	Irix	Linux	macOS	NetBSD	OpenBSD	Solaris	Tru64 UNIX	Windows
ATM	❓	❓	❓	❓	✅	❌	❓	❓	✅	❓	❓
Bluetooth	❌	❌	❌	❌	✅	❌	❌	❌	❌	❌	❌
CiscoHDLC	❓	✅	❓	❓	✅	❓	✅	✅	❓	❓	❓
Ethernet	✅	✅	✅	✅	✅	✅	✅	✅	✅	✅	✅
FDDI (Fiber Distributed Data Interface)	❓	❓	❓	❓	✅	❌	❓	❓	✅	❓	❓
Frame Relay	❓	❓	❌	❌	✅	❌	❓	❓	❌	❌	❌
IrDA (Infrared Data Association)	❌	❌	❌	❌	✅	❌	❌	❌	❌	❌	❌
PPP	❓	❓	❓	❓	✅	✅	❓	❓	❌	❓	✅
TokenRing	✅	✅	❓	❌	✅	❌	✅	✅	✅	❓	✅
USB	❌	❌	❌	❌	✅	❌	❌	❌	❌	❌	❌
WLAN	❓	✅	❓	❓	✅	✅	✅	✅	❓	❓	✅
Loopback (virtual)	❓	✅	❌	❓	✅	✅	✅	✅	❌	✅	N/A (Not Applicable)
VLAN Tags (virtual)	✅	✅	✅	❓	✅	✅	✅	✅	✅	✅	✅

Network different kinds that Wireshark endorses through multiple platforms in Table 1.

In the table, the symbols likely represent the following meanings:

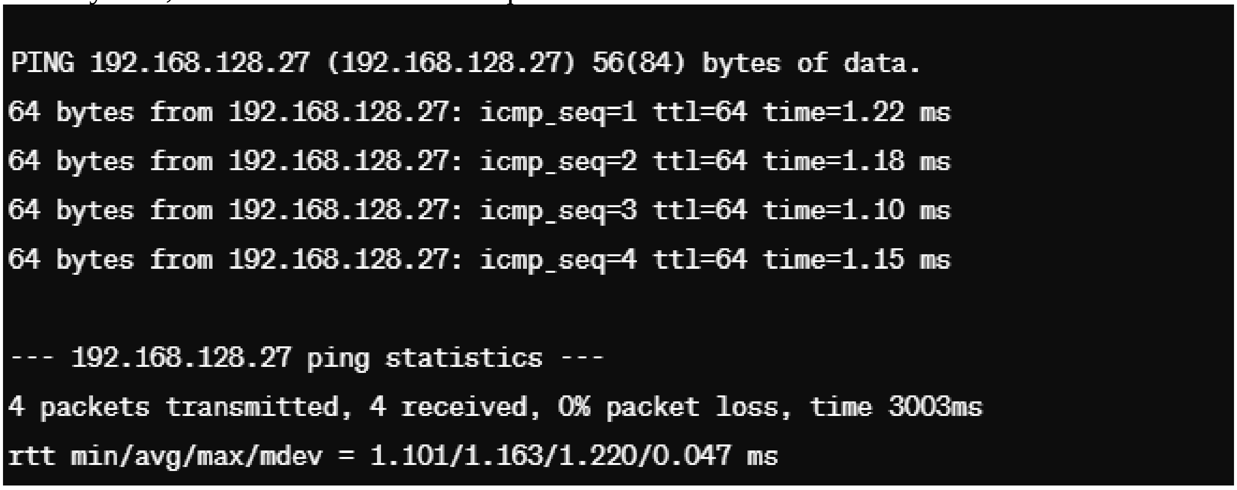
- ✅ **(Checkmark)**: The feature or physical interface is supported by the operating system.
- ❌ **(Cross)**: The feature or physical interface is not supported by the operating system.

- ❓ **(Question Mark)**: Uncertainty or limited information on whether the feature or interface is supported by the operating system.
- N/A (Not Applicable): The feature or interface is not relevant or applicable to that operating system.

These symbols help quickly identify the compatibility of different network interfaces with various operating systems.

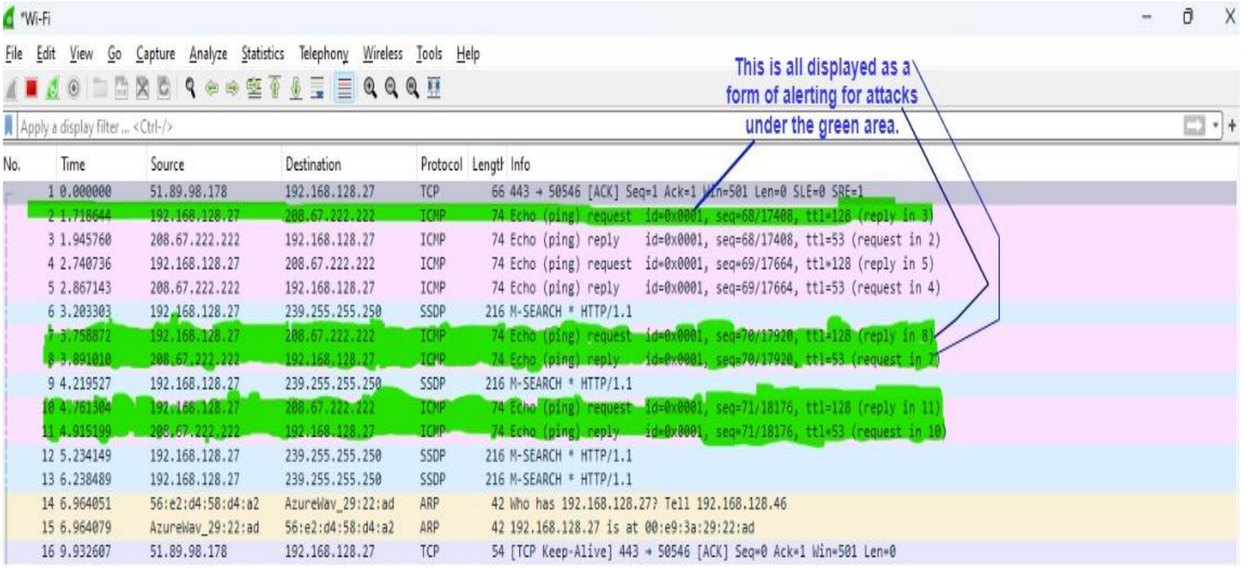
III. An Intrusion Detection System (IDS) monitors and analyzes network packets to identify suspicious activity and potential security breaches, as illustrated in Figures 12-15. By examining traffic patterns and detecting anomalies, the IDS plays a crucial role in enhancing network security, enabling timely alerts and responses to potential threats, thereby safeguarding the integrity of the network environment.

During the monitoring period, a notable event was observed where the IP address 192.168.128.27 (identified as the attacker) was sending an excessive volume of ICMP Echo Request packets (ping requests) to the target IP address 208.67.222.222 (the victim) as depicted in the Figure 13. This behavior suggests a potential Denial of Service (DoS) or Distributed Denial of Service (DDoS) attack. Such a high frequency of requests can significantly impair the performance and responsiveness of the victim's system, as discussed in the subsequent sections.



```
PING 192.168.128.27 (192.168.128.27) 56(84) bytes of data.
64 bytes from 192.168.128.27: icmp_seq=1 ttl=64 time=1.22 ms
64 bytes from 192.168.128.27: icmp_seq=2 ttl=64 time=1.18 ms
64 bytes from 192.168.128.27: icmp_seq=3 ttl=64 time=1.10 ms
64 bytes from 192.168.128.27: icmp_seq=4 ttl=64 time=1.15 ms

--- 192.168.128.27 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 1.101/1.163/1.220/0.047 ms
```

Figure 12: Ping statistics for attacker and victim IP addresses

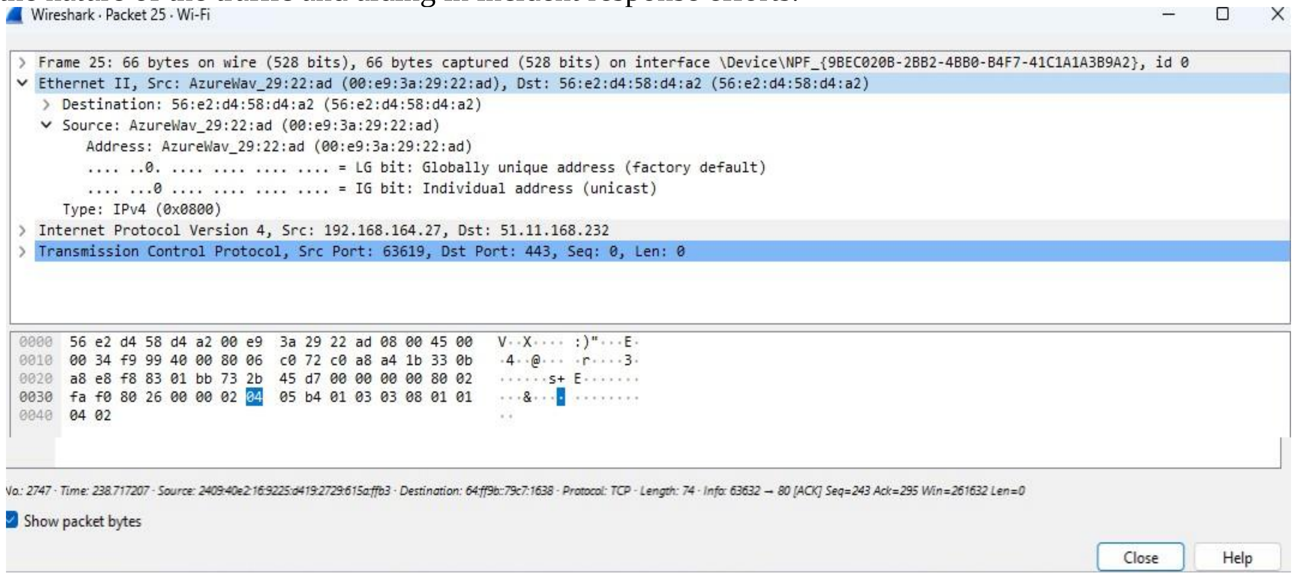

*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

This is all displayed as a form of alerting for attacks under the green area.

No.	Time	Source	Destination	Protocol	Lengtł	Info
1	0.000000	51.89.98.178	192.168.128.27	TCP	66	443 → 50546 [ACK] Seq=1 Ack=1 Win=501 Len=0 SLE=0 SRE=1
2	1.718644	192.168.128.27	208.67.222.222	ICMP	74	Echo (ping) request id=0x0001, seq=68/17408, ttl=128 (reply in 3)
3	1.945760	208.67.222.222	192.168.128.27	ICMP	74	Echo (ping) reply id=0x0001, seq=68/17408, ttl=53 (request in 2)
4	2.740736	192.168.128.27	208.67.222.222	ICMP	74	Echo (ping) request id=0x0001, seq=69/17664, ttl=128 (reply in 5)
5	2.867143	208.67.222.222	192.168.128.27	ICMP	74	Echo (ping) reply id=0x0001, seq=69/17664, ttl=53 (request in 4)
6	3.203303	192.168.128.27	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
7	3.758872	192.168.128.27	208.67.222.222	ICMP	74	Echo (ping) request id=0x0001, seq=70/17920, ttl=128 (reply in 8)
8	3.891010	208.67.222.222	192.168.128.27	ICMP	74	Echo (ping) reply id=0x0001, seq=70/17920, ttl=53 (request in 7)
9	4.219527	192.168.128.27	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
10	4.761304	192.168.128.27	208.67.222.222	ICMP	74	Echo (ping) request id=0x0001, seq=71/18176, ttl=128 (reply in 11)
11	4.915199	208.67.222.222	192.168.128.27	ICMP	74	Echo (ping) reply id=0x0001, seq=71/18176, ttl=53 (request in 10)
12	5.234149	192.168.128.27	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
13	6.238489	192.168.128.27	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
14	6.964051	56:e2:d4:58:d4:a2	AzureWav_29:22:ad	ARP	42	Who has 192.168.128.27? Tell 192.168.128.46
15	6.964079	AzureWav_29:22:ad	56:e2:d4:58:d4:a2	ARP	42	192.168.128.27 is at 00:e9:3a:29:22:ad
16	9.932607	51.89.98.178	192.168.128.27	TCP	54	[TCP Keep-Alive] 443 → 50546 [ACK] Seq=0 Ack=1 Win=501 Len=0

Figure 13: Ping requests are frequently sent to the victim's IP address 208.67.222.222 from the attacker's IP address 192.168.128.27

Figure 13 displays packet capture data showing network activity, highlighting a significant number of ICMP Echo Requests (ping requests) from the attacker (192.168.128.27) to the victim (208.67.222.222), indicative of a potential DoS or DDoS attack. The highlighted area signifies alerts for anomalous traffic patterns, allowing for rapid identification and response. Various protocols, including TCP and SSDP (Simple Service Discovery Protocol), are depicted, providing insights into the nature of the traffic and aiding in incident response efforts.

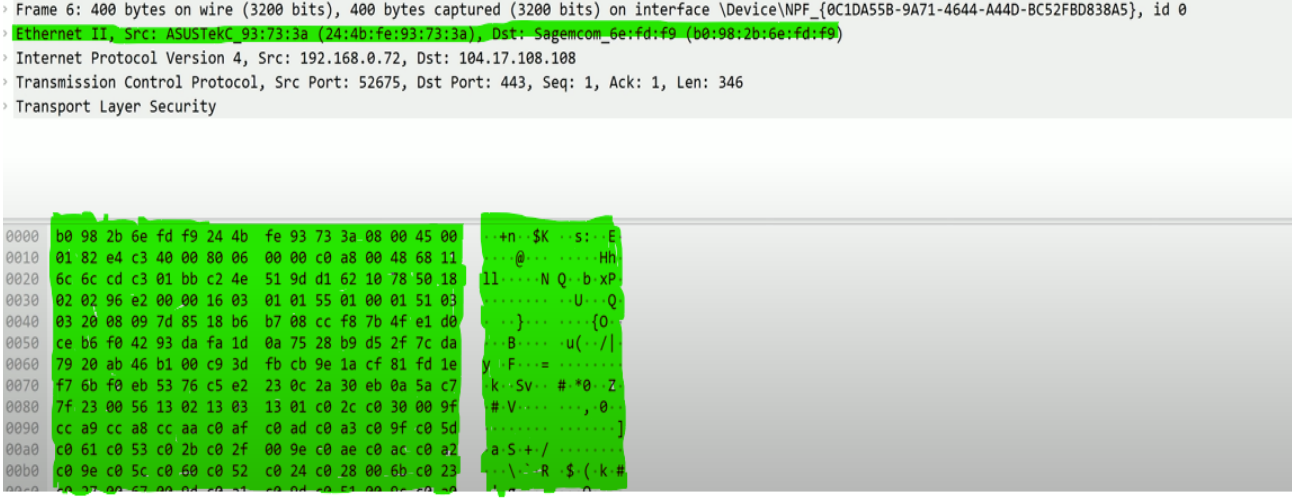


```
> Frame 25: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{9BEC020B-2BB2-4BB0-B4F7-41C1A1A3B9A2}, id 0
v Ethernet II, Src: AzureWav_29:22:ad (00:e9:3a:29:22:ad), Dst: 56:e2:d4:58:d4:a2 (56:e2:d4:58:d4:a2)
  > Destination: 56:e2:d4:58:d4:a2 (56:e2:d4:58:d4:a2)
  v Source: AzureWav_29:22:ad (00:e9:3a:29:22:ad)
      Address: AzureWav_29:22:ad (00:e9:3a:29:22:ad)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 192.168.164.27, Dst: 51.11.168.232
> Transmission Control Protocol, Src Port: 63619, Dst Port: 443, Seq: 0, Len: 0

0000  56 e2 d4 58 d4 a2 00 e9  3a 29 22 ad 08 00 45 00   V··X···· :)"···E·
0010  00 34 f9 99 40 00 80 06  c0 72 c0 a8 a4 1b 33 0b   ·4··@··· ·r····3·
0020  a8 e8 f8 83 01 bb 73 2b  45 d7 00 00 00 00 80 02   ······s+ E·······
0030  fa f0 80 26 00 00 02 04  05 b4 01 03 03 08 01 01   ···&···· ········
0040  04 02                                              ··

No.: 2747 · Time: 238.717207 · Source: 2409:40e2:16:9225:d419:2729:615a:ffb3 · Destination: 64:ff9b::79c7:1638 · Protocol: TCP · Length: 74 · Info: 63632 → 80 [ACK] Seq=243 Ack=295 Win=261632 Len=0
```

☑ Show packet bytes

Figure 14: Evaluation of data transmitted and received over a computer network

Figure 14 shows the process or analysis of network data flow, focusing on the transmission and reception of data packets. Typically, such a figure would represent key metrics, such as the volume of data transmitted and received, packet types, protocols used (like TCP/UDP), and other performance indicators. This Wireshark capture shows a TCP packet with Ethernet II details. The source MAC is Azurewav_29:22, and the destination MAC is 56:62:d4:58:d4:82. The IPv4 layer specifies source IP 192.168.164.27 and destination IP 51.11.168.232. The TCP layer reveals source port 65819 and destination port 443, indicating HTTPS communication.

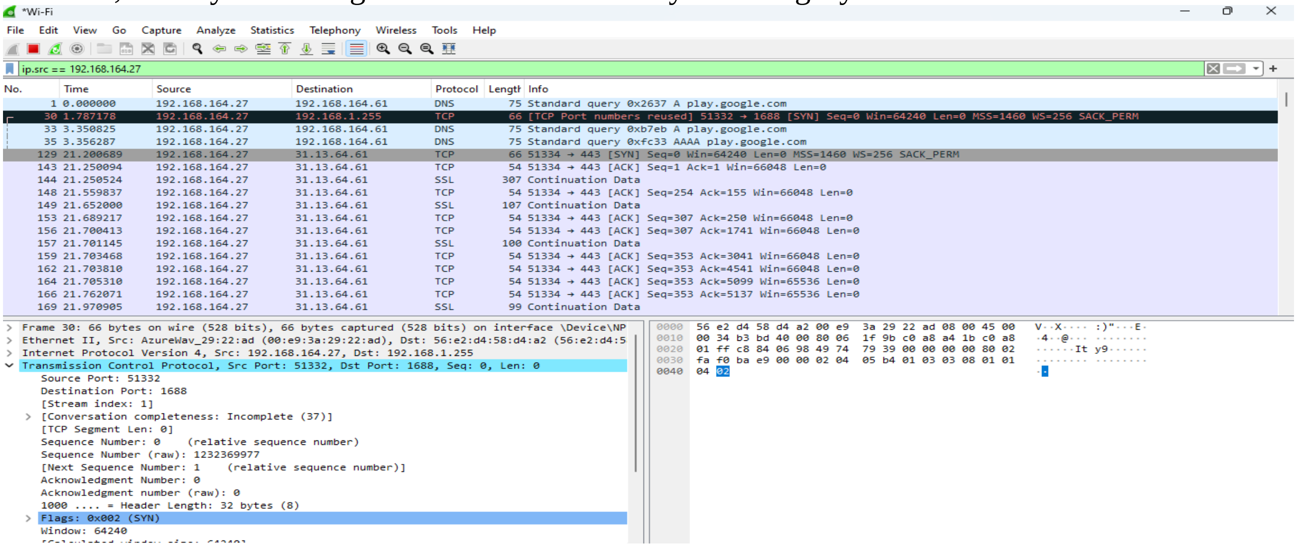


```
> Frame 6: 400 bytes on wire (3200 bits), 400 bytes captured (3200 bits) on interface \Device\NPF_{0C1DA55B-9A71-4644-A44D-BC52FBD838A5}, id 0
> Ethernet II, Src: ASUSTekC_93:73:3a (24:4b:fe:93:73:3a), Dst: Sagemcom_6e:fd:f9 (b0:98:2b:6e:fd:f9)
> Internet Protocol Version 4, Src: 192.168.0.72, Dst: 104.17.108.108
> Transmission Control Protocol, Src Port: 52675, Dst Port: 443, Seq: 1, Ack: 1, Len: 346
> Transport Layer Security

0000  b0 98 2b 6e fd f9 24 4b  fe 93 73 3a 08 00 45 00   ··+n··$K ··s:··E·
0010  01 82 e4 c3 40 00 80 06  00 00 c0 a8 00 48 68 11   ····@··· ·····Hh·
0020  6c 6c cd c3 01 bb c2 4e  51 9d d1 62 10 78 50 18   ll·····N Q··b·xP·
0030  02 02 96 e2 00 00 16 03  01 01 55 01 00 01 51 03   ········ ··U···Q·
0040  03 20 08 09 7d 85 18 b6  b7 08 cc f8 7b 4f e1 d0   · ··}··· ····{O··
0050  ce b6 f0 42 93 da fa 1d  0a 75 28 b9 d5 2f 7c da   ···B···· ·u(··/|·
0060  79 20 ab 46 b1 00 c9 3d  fb cb 9e 1a cf 81 fd 1e   y ·F···= ········
0070  f7 6b f0 eb 53 76 c5 e2  23 0c 2a 30 eb 0a 5a c7   ·k··Sv·· #·*0··Z·
0080  7f 23 00 56 13 02 13 03  13 01 c0 2c c0 30 00 9f   ·#·V···· ···,·0··
0090  cc a9 cc a8 cc aa c0 af  c0 ad c0 a3 c0 9f c0 5d   ········ ·······]
00a0  c0 61 c0 53 c0 2b c0 2f  00 9e c0 ae c0 ac c0 a2   ·a·S·+·/ ········
00b0  c0 9e c0 5c c0 60 c0 52  c0 24 c0 28 00 6b c0 23   ···\·`·R ·$·(·k·#
```

Figure 15: The Evaluation of a LAN network intrusion

Figure 15 presents an analysis of a Local Area Network (LAN) for potential intrusions. It showcases various traffic patterns, highlighting anomalies indicative of unauthorized access attempts or malicious activities. The evaluation includes metrics such as packet counts, connection attempts, and alert triggers, providing insights into the network's security posture. By monitoring these parameters, network administrators can effectively identify vulnerabilities and respond to threats, enhancing the overall security and integrity of the LAN environment.

System operators receive alerts regarding suspicious events or attacks through an Intrusion Detection System (IDS), as illustrated in Figures 16-18. This system continuously monitors network traffic and analyzes data patterns to identify potential threats, enabling timely notification and response to incidents, thereby enhancing overall network security and integrity.

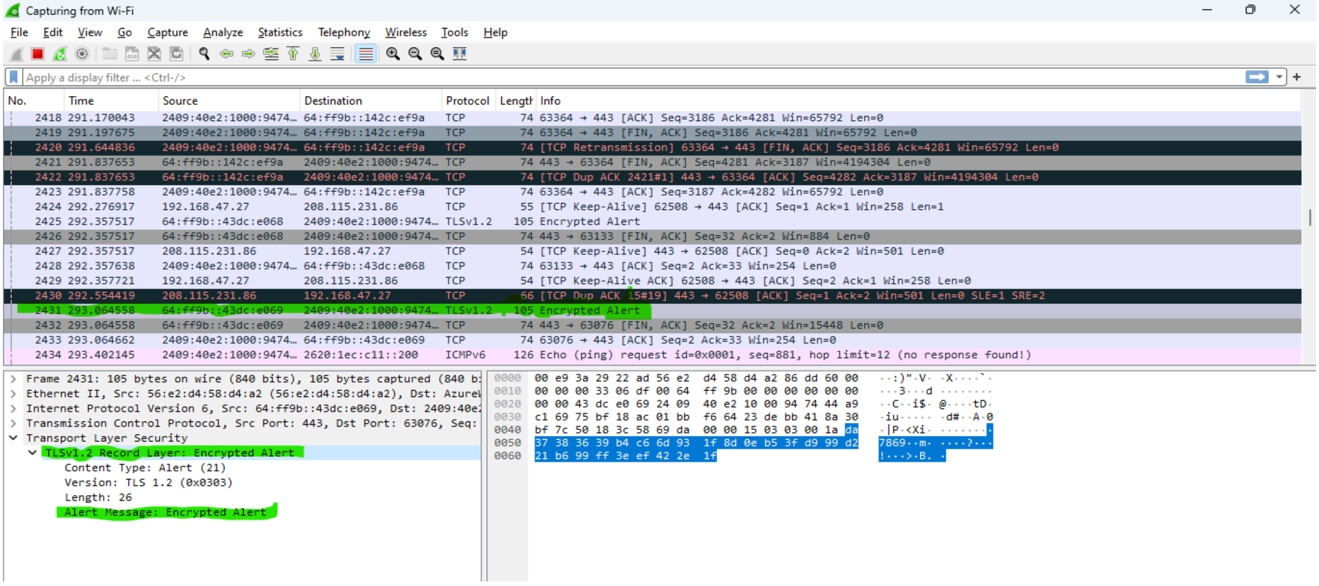


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
ip.src == 192.168.164.27

No.  Time       Source          Destination     Protocol Length Info
  1 0.000000   192.168.164.27  192.168.164.61  DNS       75 Standard query 0x2637 A play.google.com
 30 1.787178   192.168.164.27  192.168.1.255   TCP       66 [TCP Port numbers reused] 51332 → 1688 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
 33 3.350825   192.168.164.27  192.168.164.61  DNS       75 Standard query 0xb7eb A play.google.com
 35 3.356287   192.168.164.27  192.168.164.61  DNS       75 Standard query 0xfc33 AAAA play.google.com
129 21.200689  192.168.164.27  31.13.64.61     TCP       66 51334 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
143 21.250094  192.168.164.27  31.13.64.61     TCP       54 51334 → 443 [ACK] Seq=1 Ack=1 Win=66048 Len=0
144 21.250524  192.168.164.27  31.13.64.61     SSL      307 Continuation Data
148 21.559837  192.168.164.27  31.13.64.61     TCP       54 51334 → 443 [ACK] Seq=254 Ack=155 Win=66048 Len=0
149 21.652000  192.168.164.27  31.13.64.61     SSL      107 Continuation Data
153 21.689217  192.168.164.27  31.13.64.61     TCP       54 51334 → 443 [ACK] Seq=307 Ack=250 Win=66048 Len=0
156 21.700413  192.168.164.27  31.13.64.61     TCP       54 51334 → 443 [ACK] Seq=307 Ack=1741 Win=66048 Len=0
157 21.701145  192.168.164.27  31.13.64.61     SSL      100 Continuation Data
159 21.703468  192.168.164.27  31.13.64.61     TCP       54 51334 → 443 [ACK] Seq=353 Ack=3041 Win=66048 Len=0
162 21.703810  192.168.164.27  31.13.64.61     TCP       54 51334 → 443 [ACK] Seq=353 Ack=4541 Win=66048 Len=0
164 21.705310  192.168.164.27  31.13.64.61     TCP       54 51334 → 443 [ACK] Seq=353 Ack=5099 Win=65536 Len=0
166 21.762071  192.168.164.27  31.13.64.61     TCP       54 51334 → 443 [ACK] Seq=353 Ack=5137 Win=65536 Len=0
169 21.970905  192.168.164.27  31.13.64.61     SSL       99 Continuation Data

> Frame 30: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NP
> Ethernet II, Src: AzureWav_29:22:ad (00:e9:3a:29:22:ad), Dst: 56:e2:d4:58:d4:a2 (56:e2:d4:5
> Internet Protocol Version 4, Src: 192.168.164.27, Dst: 192.168.1.255
v Transmission Control Protocol, Src Port: 51332, Dst Port: 1688, Seq: 0, Len: 0
    Source Port: 51332
    Destination Port: 1688
    [Stream index: 1]
  > [Conversation completeness: Incomplete (37)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 1232369977
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x002 (SYN)
    Window: 64240

0000  56 e2 d4 58 d4 a2 00 e9  3a 29 22 ad 08 00 45 00   V··X···· :)"···E·
0010  00 34 b3 bd 40 00 80 06  1f 9b c0 a8 a4 1b c0 a8   ·4··@··· ········
0020  01 ff c8 84 06 98 49 74  79 39 00 00 00 00 80 02   ······It y9······
0030  fa f0 ba e9 00 00 02 04  05 b4 01 03 03 08 01 01   ········ ········
0040  04 02                                              ··
```

Figure 16: An overview of linked networks coupled with specific information regarding each protocol

Figure 16 provides a detailed overview of interconnected networks, highlighting the specific protocols employed across each link. It includes data on protocols such as TCP/IP, UDP, and HTTP, detailing their roles in data transmission, reliability, and session management. Additionally, the figure illustrates network topologies and the flow of information between nodes, offering insights into network architecture and communication efficiency. Understanding these protocols is crucial for optimizing performance and enhancing security measures within the network infrastructure.

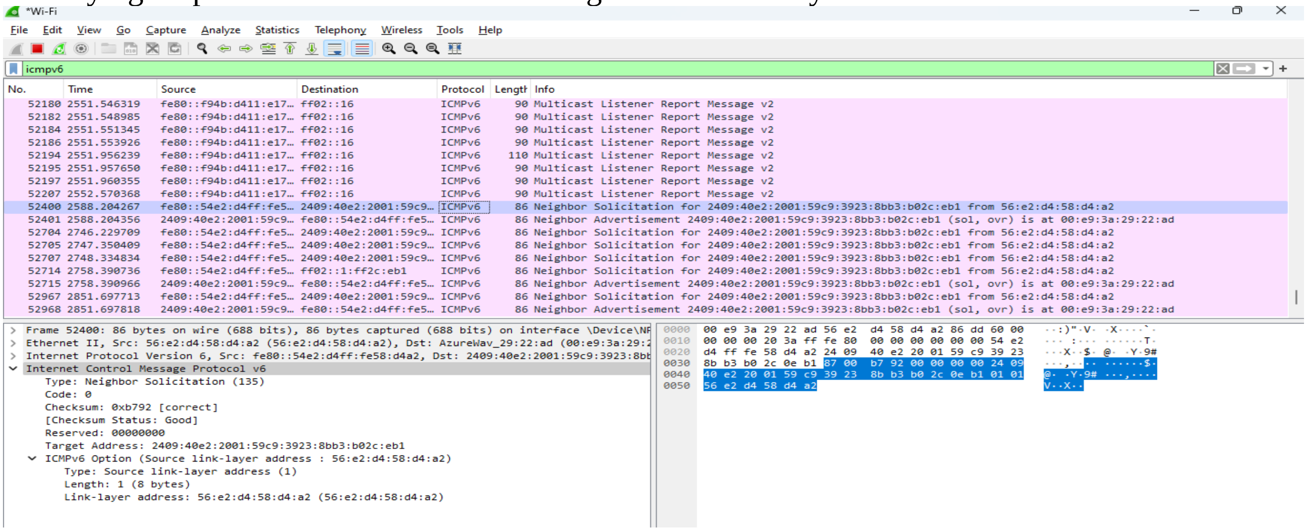


Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
Apply a display filter ... <Ctrl-/>

No.   Time        Source                Destination           Protocol Length Info
2418 291.170043  2409:40e2:1000:9474…  64:ff9b::142c:ef9a    TCP       74 63364 → 443 [ACK] Seq=3186 Ack=4281 Win=65792 Len=0
2419 291.197675  2409:40e2:1000:9474…  64:ff9b::142c:ef9a    TCP       74 63364 → 443 [FIN, ACK] Seq=3186 Ack=4281 Win=65792 Len=0
2420 291.644836  2409:40e2:1000:9474…  64:ff9b::142c:ef9a    TCP       74 [TCP Retransmission] 63364 → 443 [FIN, ACK] Seq=3186 Ack=4281 Win=65792 Len=0
2421 291.837653  64:ff9b::142c:ef9a    2409:40e2:1000:9474…  TCP       74 443 → 63364 [FIN, ACK] Seq=4281 Ack=3187 Win=4194304 Len=0
2422 291.837653  64:ff9b::142c:ef9a    2409:40e2:1000:9474…  TCP       74 [TCP Dup ACK 2421#1] 443 → 63364 [ACK] Seq=4282 Ack=3187 Win=4194304 Len=0
2423 291.837758  2409:40e2:1000:9474…  64:ff9b::142c:ef9a    TCP       74 63364 → 443 [ACK] Seq=3187 Ack=4282 Win=65792 Len=0
2424 292.276917  192.168.47.27         208.115.231.86        TCP       55 [TCP Keep-Alive] 62508 → 443 [ACK] Seq=1 Ack=1 Win=258 Len=1
2425 292.357517  64:ff9b::43dc:e068    2409:40e2:1000:9474…  TLSv1.2  105 Encrypted Alert
2426 292.357517  64:ff9b::43dc:e068    2409:40e2:1000:9474…  TCP       74 443 → 63133 [FIN, ACK] Seq=32 Ack=2 Win=884 Len=0
2427 292.357517  208.115.231.86        192.168.47.27         TCP       54 [TCP Keep-Alive] 443 → 62508 [ACK] Seq=0 Ack=2 Win=501 Len=0
2428 292.357638  2409:40e2:1000:9474…  64:ff9b::43dc:e068    TCP       74 63133 → 443 [ACK] Seq=2 Ack=33 Win=254 Len=0
2429 292.357721  192.168.47.27         208.115.231.86        TCP       54 [TCP Keep-Alive ACK] 62508 → 443 [ACK] Seq=2 Ack=1 Win=258 Len=0
2430 292.554419  208.115.231.86        192.168.47.27         TCP       66 [TCP Dup ACK 15#19] 443 → 62508 [ACK] Seq=1 Ack=2 Win=501 Len=0 SLE=1 SRE=2
2431 293.064558  64:ff9b::43dc:e069    2409:40e2:1000:9474…  TLSv1.2  105 Encrypted Alert
2432 293.064558  64:ff9b::43dc:e069    2409:40e2:1000:9474…  TCP       74 443 → 63076 [FIN, ACK] Seq=32 Ack=2 Win=15448 Len=0
2433 293.064662  2409:40e2:1000:9474…  64:ff9b::43dc:e069    TCP       74 63076 → 443 [ACK] Seq=2 Ack=33 Win=254 Len=0
2434 293.402145  2409:40e2:1000:9474…  2620:1ec:c11::200     ICMPv6   126 Echo (ping) request id=0x0001, seq=881, hop limit=12 (no response found!)

> Frame 2431: 105 bytes on wire (840 bits), 105 bytes captured (840 b
> Ethernet II, Src: 56:e2:d4:58:d4:a2 (56:e2:d4:58:d4:a2), Dst: Azure
> Internet Protocol Version 6, Src: 64:ff9b::43dc:e069, Dst: 2409:40e
> Transmission Control Protocol, Src Port: 443, Dst Port: 63076, Seq:
v Transport Layer Security
  v TLSv1.2 Record Layer: Encrypted Alert
      Content Type: Alert (21)
      Version: TLS 1.2 (0x0303)
      Length: 26
      Alert Message: Encrypted Alert

0000  00 e9 3a 29 22 ad 56 e2  d4 58 d4 a2 86 dd 60 00   ··:)"·V· ·X····`·
0010  00 00 00 33 06 df 00 64  ff 9b 00 00 00 00 00 00   ···3···d ········
0020  00 00 43 dc e0 69 24 09  40 e2 10 00 94 74 44 a9   ··C··i$· @····tD·
0030  c1 69 75 bf 18 ac 01 bb  f6 64 23 de bb 41 8a 30   ·iu····· ·d#··A·0
0040  bf 7c 50 18 3c 58 69 da  00 00 15 03 03 00 1a da   ·|P·<Xi· ········
0050  37 38 36 39 b4 c6 6d 93  1f 8d 0e b5 3f d9 99 d2   7869··m· ····?···
0060  21 b6 99 ff 3e ef 42 2e  1f                        !···>·B. ·
```

Figure 17: Monitoring data for Intrusion

Figure 17 showcases captured network traffic, highlighting interactions between various IP addresses and alerts for potential anomalies. Notably, Transport Layer Security (TLS) alerts indicate encrypted communications designed to safeguard data confidentiality. The displayed TCP protocol details include flags that reveal connection states, while the hex view at the bottom provides a raw data representation of packet payloads, essential for in-depth analysis. This information is crucial for identifying suspicious activities and enhancing network security.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

icmpv6

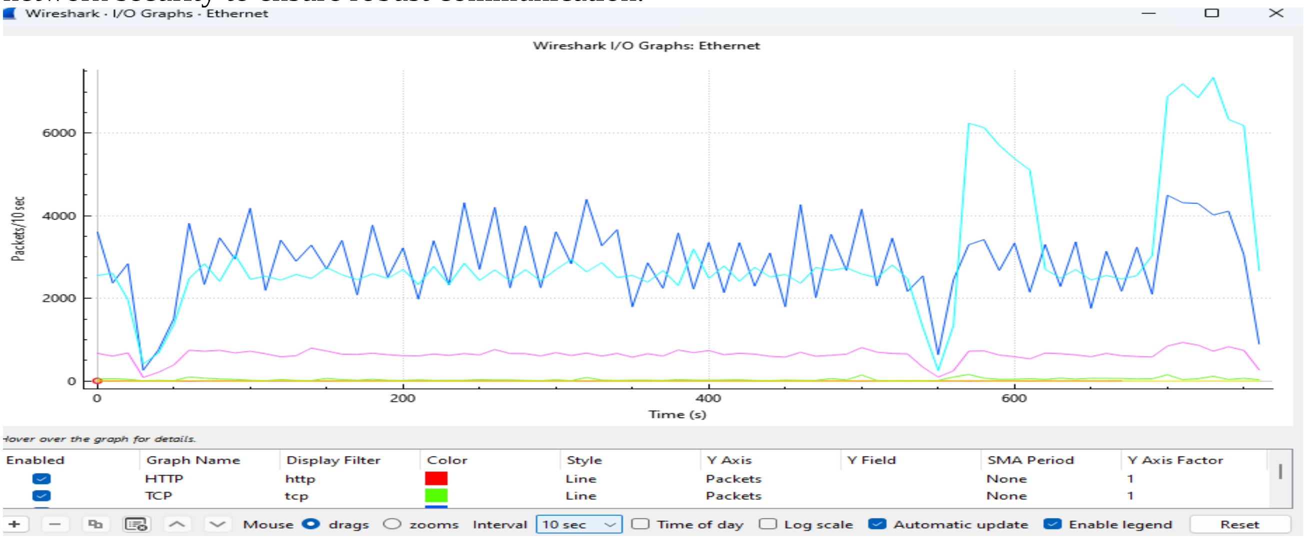
```
No.   Time        Source                Destination           Protocol Lengtl Info
52180 2551.546319 fe80::f94b:d411:e17… ff02::16              ICMPv6    90 Multicast Listener Report Message v2
52182 2551.548985 fe80::f94b:d411:e17… ff02::16              ICMPv6    90 Multicast Listener Report Message v2
52184 2551.551345 fe80::f94b:d411:e17… ff02::16              ICMPv6    90 Multicast Listener Report Message v2
52186 2551.553926 fe80::f94b:d411:e17… ff02::16              ICMPv6    90 Multicast Listener Report Message v2
52194 2551.956239 fe80::f94b:d411:e17… ff02::16              ICMPv6   110 Multicast Listener Report Message v2
52195 2551.957650 fe80::f94b:d411:e17… ff02::16              ICMPv6    90 Multicast Listener Report Message v2
52197 2551.960355 fe80::f94b:d411:e17… ff02::16              ICMPv6    90 Multicast Listener Report Message v2
52207 2552.570368 fe80::f94b:d411:e17… ff02::16              ICMPv6    90 Multicast Listener Report Message v2
52400 2588.204267 fe80::54e2:d4ff:fe5… 2409:40e2:2001:59c9… ICMPv6    86 Neighbor Solicitation for 2409:40e2:2001:59c9:3923:8bb3:b02c:eb1 from 56:e2:d4:58:d4:a2
52401 2588.204356 2409:40e2:2001:59c9… fe80::54e2:d4ff:fe5… ICMPv6    86 Neighbor Advertisement 2409:40e2:2001:59c9:3923:8bb3:b02c:eb1 (sol, ovr) is at 00:e9:3a:29:22:ad
52704 2746.229709 fe80::54e2:d4ff:fe5… 2409:40e2:2001:59c9… ICMPv6    86 Neighbor Solicitation for 2409:40e2:2001:59c9:3923:8bb3:b02c:eb1 from 56:e2:d4:58:d4:a2
52705 2747.350409 fe80::54e2:d4ff:fe5… 2409:40e2:2001:59c9… ICMPv6    86 Neighbor Solicitation for 2409:40e2:2001:59c9:3923:8bb3:b02c:eb1 from 56:e2:d4:58:d4:a2
52707 2748.334834 fe80::54e2:d4ff:fe5… 2409:40e2:2001:59c9… ICMPv6    86 Neighbor Solicitation for 2409:40e2:2001:59c9:3923:8bb3:b02c:eb1 from 56:e2:d4:58:d4:a2
52714 2758.390736 fe80::54e2:d4ff:fe5… ff02::1:ff2c:eb1      ICMPv6    86 Neighbor Solicitation for 2409:40e2:2001:59c9:3923:8bb3:b02c:eb1 from 56:e2:d4:58:d4:a2
52715 2758.390966 2409:40e2:2001:59c9… fe80::54e2:d4ff:fe5… ICMPv6    86 Neighbor Advertisement 2409:40e2:2001:59c9:3923:8bb3:b02c:eb1 (sol, ovr) is at 00:e9:3a:29:22:ad
52967 2851.697713 fe80::54e2:d4ff:fe5… 2409:40e2:2001:59c9… ICMPv6    86 Neighbor Solicitation for 2409:40e2:2001:59c9:3923:8bb3:b02c:eb1 from 56:e2:d4:58:d4:a2
52968 2851.697818 2409:40e2:2001:59c9… fe80::54e2:d4ff:fe5… ICMPv6    86 Neighbor Advertisement 2409:40e2:2001:59c9:3923:8bb3:b02c:eb1 (sol, ovr) is at 00:e9:3a:29:22:ad
```

```
> Frame 52400: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface \Device\NF
> Ethernet II, Src: 56:e2:d4:58:d4:a2 (56:e2:d4:58:d4:a2), Dst: AzureWav_29:22:ad (00:e9:3a:29:2
> Internet Protocol Version 6, Src: fe80::54e2:d4ff:fe58:d4a2, Dst: 2409:40e2:2001:59c9:3923:8bb
v Internet Control Message Protocol v6
    Type: Neighbor Solicitation (135)
    Code: 0
    Checksum: 0xb792 [correct]
    [Checksum Status: Good]
    Reserved: 00000000
    Target Address: 2409:40e2:2001:59c9:3923:8bb3:b02c:eb1
  v ICMPv6 Option (Source link-layer address : 56:e2:d4:58:d4:a2)
      Type: Source link-layer address (1)
      Length: 1 (8 bytes)
      Link-layer address: 56:e2:d4:58:d4:a2 (56:e2:d4:58:d4:a2)
```

```
0000  00 e9 3a 29 22 ad 56 e2  d4 58 d4 a2 86 dd 60 00   ··:)"·V· ·X····`·
0010  00 00 00 20 3a ff fe 80  00 00 00 00 00 00 54 e2   ··· :··· ······T·
0020  d4 ff fe 58 d4 a2 24 09  40 e2 20 01 59 c9 39 23   ···X··$· @· ·Y·9#
0030  8b b3 b0 2c 0e b1 87 00  b7 92 00 00 00 00 24 09   ···,···· ······$·
0040  40 e2 20 01 59 c9 39 23  8b b3 b0 2c 0e b1 01 01   @· ·Y·9# ···,····
0050  56 e2 d4 58 d4 a2                                  V··X··
```

Figure 18: Evaluating network activity using Wireshark

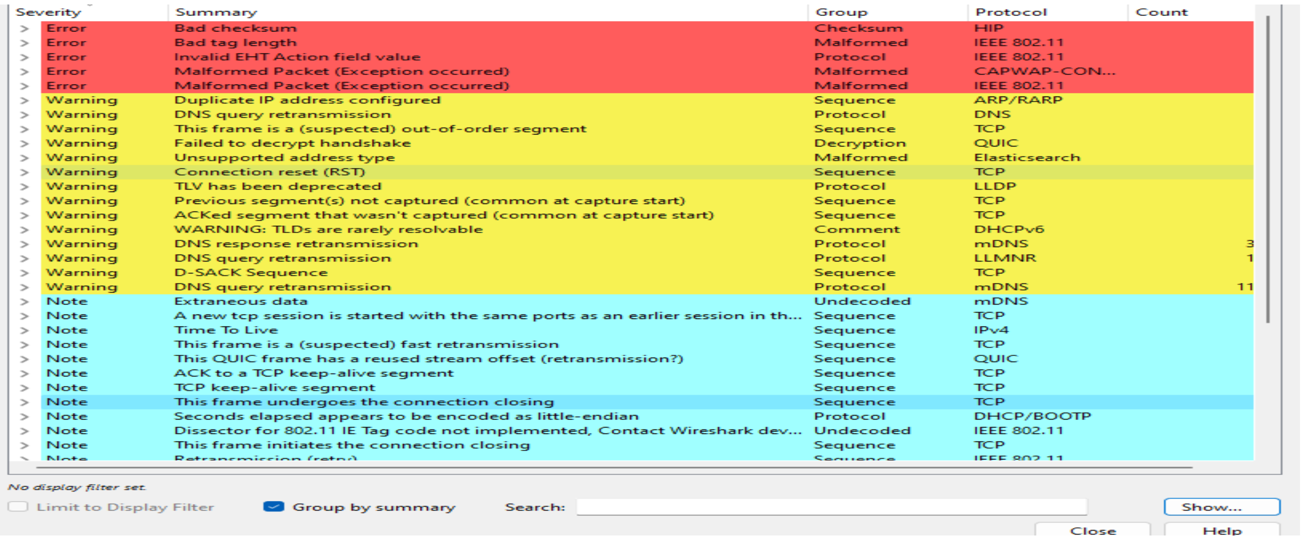
Figure 18 illustrates the analysis of network activity captured by Wireshark, showcasing various protocols and traffic patterns. It highlights key metrics such as packet counts, response times, and connection statuses, providing detailed insights into network performance. This visual representation helps identify anomalies, potential bottlenecks, and security threats. By examining this data, network administrators can improve troubleshooting efforts, optimize performance, and strengthen overall network security to ensure robust communication.



Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☑	HTTP	http	🟥	Line	Packets		None	1
☑	TCP	tcp	🟩	Line	Packets		None	1

Figure 19: I/O graph of transmitted data packets that were captured

Figure 19 shows a Wireshark I/O Graph depicting network traffic over time, specifically for Ethernet traffic. It includes separate lines for different protocols like HTTP (in red), TCP (in green), and other unidentified protocols in blue. The Y-axis represents the packet rate (packets/sec), and the X-axis shows time in seconds. The graph tracks the packet volume over a set interval of 10 seconds. The visible spikes and dips indicate fluctuations in network activity, which may suggest moments of high traffic or potential network issues being analyzed.



Severity	Summary	Group	Protocol	Count
> Error	Bad checksum	Checksum	HIP	
> Error	Bad tag length	Malformed	IEEE 802.11	
> Error	Invalid EHT Action field value	Protocol	IEEE 802.11	
> Error	Malformed Packet (Exception occurred)	Malformed	CAPWAP-CON...	
> Error	Malformed Packet (Exception occurred)	Malformed	IEEE 802.11	
> Warning	Duplicate IP address configured	Sequence	ARP/RARP	
> Warning	DNS query retransmission	Protocol	DNS	
> Warning	This frame is a (suspected) out-of-order segment	Sequence	TCP	
> Warning	Failed to decrypt handshake	Decryption	QUIC	
> Warning	Unsupported address type	Malformed	Elasticsearch	
> Warning	Connection reset (RST)	Sequence	TCP	
> Warning	TLV has been deprecated	Protocol	LLDP	
> Warning	Previous segment(s) not captured (common at capture start)	Sequence	TCP	
> Warning	ACKed segment that wasn't captured (common at capture start)	Sequence	TCP	
> Warning	WARNING: TLDs are rarely resolvable	Comment	DHCPv6	
> Warning	DNS response retransmission	Protocol	mDNS	3
> Warning	DNS query retransmission	Protocol	LLMNR	1
> Warning	D-SACK Sequence	Sequence	TCP	
> Warning	DNS query retransmission	Protocol	mDNS	11
> Note	Extraneous data	Undecoded	mDNS	
> Note	A new tcp session is started with the same ports as an earlier session in th...	Sequence	TCP	
> Note	Time To Live	Sequence	IPv4	
> Note	This frame is a (suspected) fast retransmission	Sequence	TCP	
> Note	This QUIC frame has a reused stream offset (retransmission?)	Sequence	QUIC	
> Note	ACK to a TCP keep-alive segment	Sequence	TCP	
> Note	TCP keep-alive segment	Sequence	TCP	
> Note	This frame undergoes the connection closing	Sequence	TCP	
> Note	Seconds elapsed appears to be encoded as little-endian	Protocol	DHCP/BOOTP	
> Note	Dissector for 802.11 IE Tag code not implemented, Contact Wireshark dev...	Undecoded	IEEE 802.11	
> Note	This frame initiates the connection closing	Sequence	TCP	
> Note	Retransmission (retry)	Sequence	IEEE 802.11	

No display filter set.

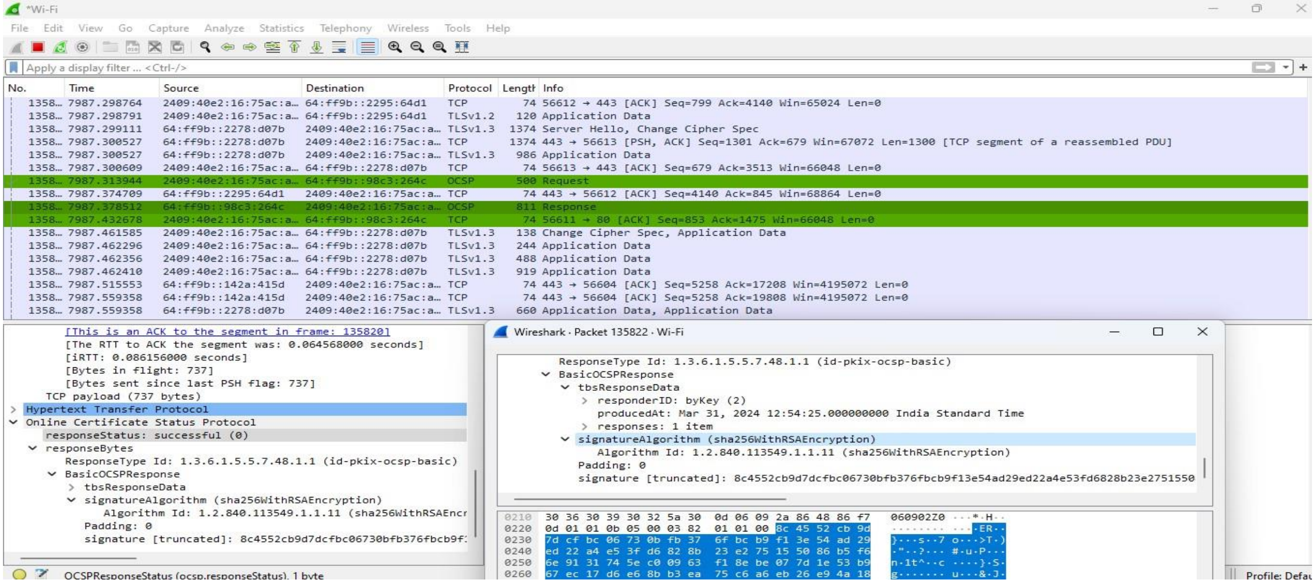
□ Limit to Display Filter ☑ Group by summary Search:

Figure 20: Expert information on captured data packets

Wireshark enhances the identification and troubleshooting of abnormal network behaviors by analyzing packet lists. It highlights suspicious packets, such as malformed ones, in red to indicate potential issues. Errors, including bad TCP packets, are flagged based on predefined coloring rules, improving visibility of protocol violations and assisting both novice and expert users in efficiently resolving network problems, as illustrated in Figure 20.

Wireshark is a network protocol analyzer designed to capture and inspect data packets traversing a network. While it cannot decrypt encrypted traffic natively, it can decrypt SSL/TLS traffic using RSA encryption if the server's private key is available. Although RSA is computationally intensive and not suitable for bulk data encryption, it is commonly used in digital signatures and key exchange mechanisms. Wireshark leverages this capability to analyze encrypted sessions, such as SSL/TLS, as shown in Figure 21.

The capture displays SSL/TLS traffic, highlighting TCP segments and handshake processes. Key details include the protocol, source, destination, and packet length. The lower window features an OCSP (Online Certificate Status Protocol) response, showing digital signature verification, responder ID, and key algorithms like SHA256WithRSAEncryption. This information indicates encrypted communication and the verification of certificate integrity during a secure session.



*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

```
No.       Time         Source                  Destination             Protocol  Length Info
  1358… 7987.298764  2409:40e2:16:75ac:a…  64:ff9b::2295:64d1    TCP        74 56612 → 443 [ACK] Seq=799 Ack=4140 Win=65024 Len=0
  1358… 7987.298791  2409:40e2:16:75ac:a…  64:ff9b::2295:64d1    TLSv1.2   120 Application Data
  1358… 7987.299111  64:ff9b::2278:d07b    2409:40e2:16:75ac:a…  TLSv1.3  1374 Server Hello, Change Cipher Spec
  1358… 7987.300527  64:ff9b::2278:d07b    2409:40e2:16:75ac:a…  TCP      1374 443 → 56613 [PSH, ACK] Seq=1301 Ack=679 Win=67072 Len=1300 [TCP segment of a reassembled PDU]
  1358… 7987.300527  64:ff9b::2278:d07b    2409:40e2:16:75ac:a…  TLSv1.3   986 Application Data
  1358… 7987.300609  2409:40e2:16:75ac:a…  64:ff9b::2278:d07b    TCP        74 56613 → 443 [ACK] Seq=679 Ack=3513 Win=66048 Len=0
  1358… 7987.313944  2409:40e2:16:75ac:a…  64:ff9b::98c3:264c    OCSP      500 Request
  1358… 7987.374709  64:ff9b::2295:64d1    2409:40e2:16:75ac:a…  TCP        74 443 → 56612 [ACK] Seq=4140 Ack=845 Win=68864 Len=0
  1358… 7987.378512  64:ff9b::98c3:264c    2409:40e2:16:75ac:a…  OCSP      811 Response
  1358… 7987.432678  2409:40e2:16:75ac:a…  64:ff9b::98c3:264c    TCP        74 56611 → 80 [ACK] Seq=853 Ack=1475 Win=66048 Len=0
  1358… 7987.461585  2409:40e2:16:75ac:a…  64:ff9b::2278:d07b    TLSv1.3   138 Change Cipher Spec, Application Data
  1358… 7987.462296  2409:40e2:16:75ac:a…  64:ff9b::2278:d07b    TLSv1.3   244 Application Data
  1358… 7987.462356  2409:40e2:16:75ac:a…  64:ff9b::2278:d07b    TLSv1.3   488 Application Data
  1358… 7987.462410  2409:40e2:16:75ac:a…  64:ff9b::2278:d07b    TLSv1.3   919 Application Data
  1358… 7987.515553  64:ff9b::142a:415d    2409:40e2:16:75ac:a…  TCP        74 443 → 56604 [ACK] Seq=5258 Ack=17208 Win=4195072 Len=0
  1358… 7987.559358  64:ff9b::142a:415d    2409:40e2:16:75ac:a…  TCP        74 443 → 56604 [ACK] Seq=5258 Ack=19808 Win=4195072 Len=0
  1358… 7987.559358  64:ff9b::2278:d07b    2409:40e2:16:75ac:a…  TLSv1.3   660 Application Data, Application Data
```

```
        [This is an ACK to the segment in frame: 135820]
        [The RTT to ACK the segment was: 0.064568000 seconds]
        [iRTT: 0.086156000 seconds]
        [Bytes in flight: 737]
        [Bytes sent since last PSH flag: 737]
     TCP payload (737 bytes)
>  Hypertext Transfer Protocol
v  Online Certificate Status Protocol
     responseStatus: successful (0)
   v responseBytes
        ResponseType Id: 1.3.6.1.5.5.7.48.1.1 (id-pkix-ocsp-basic)
      v BasicOCSPResponse
         > tbsResponseData
         v signatureAlgorithm (sha256WithRSAEncryption)
              Algorithm Id: 1.2.840.113549.1.1.11 (sha256WithRSAEncr
           Padding: 0
           signature [truncated]: 8c4552cb9d7dcfbc06730bfb376fbcb9f1

OCSPResponseStatus (ocsp.responseStatus), 1 byte
```

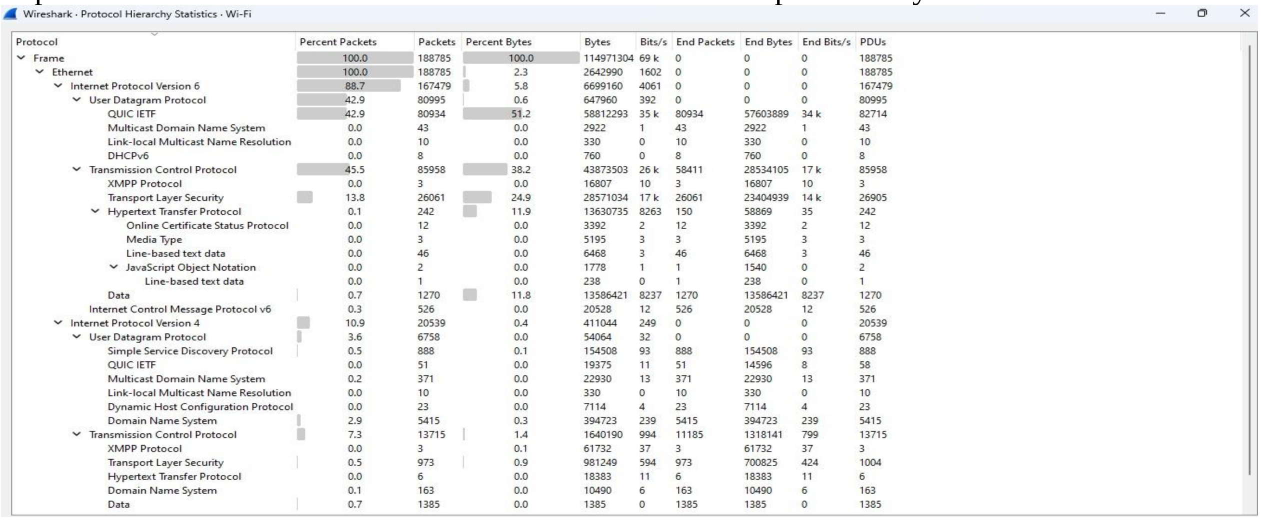
Wireshark · Packet 135822 · Wi-Fi

```
        ResponseType Id: 1.3.6.1.5.5.7.48.1.1 (id-pkix-ocsp-basic)
      v BasicOCSPResponse
         v tbsResponseData
            > responderID: byKey (2)
              producedAt: Mar 31, 2024 12:54:25.000000000 India Standard Time
            > responses: 1 item
         v signatureAlgorithm (sha256WithRSAEncryption)
              Algorithm Id: 1.2.840.113549.1.1.11 (sha256WithRSAEncryption)
           Padding: 0
           signature [truncated]: 8c4552cb9d7dcfbc06730bfb376fbcb9f13e54ad29ed22a4e53fd6828b23e2751550

0210  30 36 30 39 30 32 5a 30  0d 06 09 2a 86 48 86 f7   060902Z0 ···*·H··
0220  0d 01 01 0b 05 00 03 82  01 01 00 8c 45 52 cb 9d   ········ ···ER··
0230  7d cf bc 06 73 0b fb 37  6f bc b9 f1 3e 54 ad 29   }···s··7 o···>T·)
0240  ed 22 a4 e5 3f d6 82 8b  23 e2 75 15 50 86 b5 f6   ·"··?··· #·u·P···
0250  6e 91 31 74 5e c0 09 63  f1 8e be 07 7d 1e 53 b9   n·1t^··c ····}·S·
0260  67 ec 17 d6 e6 8b b3 ea  75 c6 a6 eb 26 e9 4a 18   g······· u···&·J·
```

Figure 21: Analyzing RSA-Encrypted Traffic with Wireshark Capture

The Wireshark protocol hierarchy statistics display an analysis of various network protocols, including Ethernet, IPv6, UDP, TCP, and application-layer protocols like TLS and OCSP. It provides insights into the percentage of packets, bytes, and bit rates handled by each protocol, helping to analyze network traffic composition. This detailed view assists in identifying the distribution and significance of protocols in the captured network data.

Wireshark's Protocol Hierarchy window provides a structured, hierarchical breakdown of network protocols observed in captured packets. It allows for an in-depth analysis of the traffic composition by detailing the relationships and dependencies between different protocol layers. This view aids in understanding how protocols interact within network flows, as illustrated in Figure 22, offering a comprehensive overview of traffic distribution across various protocol layers.

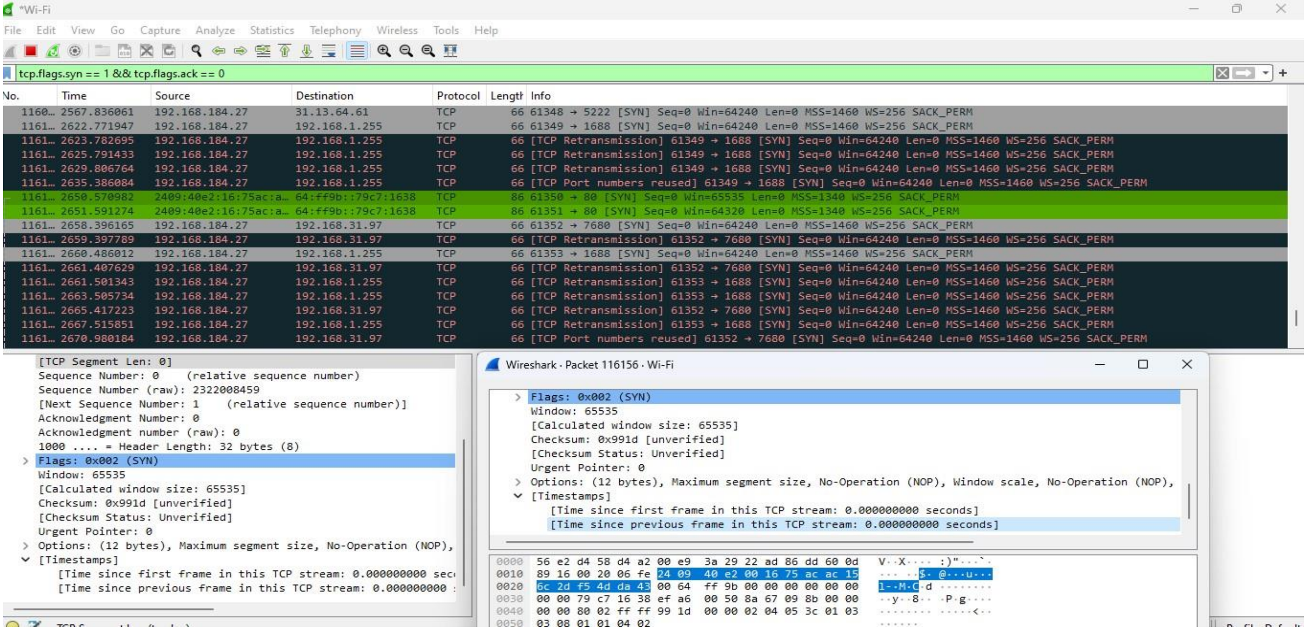


Wireshark · Protocol Hierarchy Statistics · Wi-Fi

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	188785	100.0	114971304	69 k	0	0	0	188785
Ethernet	100.0	188785	2.3	2642990	1602	0	0	0	188785
Internet Protocol Version 6	88.7	167479	5.8	6699160	4061	0	0	0	167479
User Datagram Protocol	42.9	80995	0.6	647960	392	0	0	0	80995
QUIC IETF	42.9	80934	51.2	58812293	35 k	80934	57603889	34 k	82714
Multicast Domain Name System	0.0	43	0.0	2922	1	43	2922	1	43
Link-local Multicast Name Resolution	0.0	10	0.0	330	0	10	330	0	10
DHCPv6	0.0	8	0.0	760	0	8	760	0	8
Transmission Control Protocol	45.5	85958	38.2	43873503	26 k	58411	28534105	17 k	85958
XMPP Protocol	0.0	3	0.0	16807	10	3	16807	10	3
Transport Layer Security	13.8	26061	24.9	28571034	17 k	26061	23404939	14 k	26905
Hypertext Transfer Protocol	0.1	242	11.9	13630735	8263	150	58869	35	242
Online Certificate Status Protocol	0.0	12	0.0	3392	2	12	3392	2	12
Media Type	0.0	3	0.0	5195	3	3	5195	3	3
Line-based text data	0.0	46	0.0	6468	3	46	6468	3	46
JavaScript Object Notation	0.0	2	0.0	1778	1	1	1540	0	2
Line-based text data	0.0	1	0.0	238	0	1	238	0	1
Data	0.7	1270	11.8	13586421	8237	1270	13586421	8237	1270
Internet Control Message Protocol v6	0.3	526	0.0	20528	12	526	20528	12	526
Internet Protocol Version 4	10.9	20539	0.4	411044	249	0	0	0	20539
User Datagram Protocol	3.6	6758	0.0	54064	32	0	0	0	6758
Simple Service Discovery Protocol	0.5	888	0.1	154508	93	888	154508	93	888
QUIC IETF	0.0	51	0.0	19375	11	51	14596	8	58
Multicast Domain Name System	0.2	371	0.0	22930	13	371	22930	13	371
Link-local Multicast Name Resolution	0.0	10	0.0	330	0	10	330	0	10
Dynamic Host Configuration Protocol	0.0	23	0.0	7114	4	23	7114	4	23
Domain Name System	2.9	5415	0.3	394723	239	5415	394723	239	5415
Transmission Control Protocol	7.3	13715	1.4	1640190	994	11185	1318141	799	13715
XMPP Protocol	0.0	3	0.1	61732	37	3	61732	37	3
Transport Layer Security	0.5	973	0.9	981249	594	973	700825	424	1004
Hypertext Transfer Protocol	0.0	6	0.0	18383	11	6	18383	11	6
Domain Name System	0.1	163	0.0	10490	6	163	10490	6	163
Data	0.7	1385	0.0	1385	0	1385	1385	0	1385

Figure 22: Protocol Hierarchy Analysis: Wireshark Capture Overview

Figure 23 shows a Wireshark capture of a TCP stream with detailed packet information, including sequence and acknowledgment numbers, window size, and options like No-Operation (NOP). Highlighted packets indicate TCP retransmissions, signalling potential issues such as packet loss or network latency. TCP flags—such as SYN, ACK, and FIN—are visible, reflecting the connection establishment, data transfer, and termination processes. Additionally, the hex dump provides packet payload data for in-depth analysis of each frame. TCP flags are prominently displayed in the packet details pane of Wireshark, offering crucial insights into the status and operation of TCP connection protocols.



*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.flags.syn == 1 && tcp.flags.ack == 0

```
No.      Time         Source                Destination          Protocol Length Info
1160…  2567.836061  192.168.184.27        31.13.64.61          TCP     66 61348 → 5222 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1161…  2622.771947  192.168.184.27        192.168.1.255        TCP     66 61349 → 1688 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1161…  2623.782695  192.168.184.27        192.168.1.255        TCP     66 [TCP Retransmission] 61349 → 1688 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1161…  2625.791433  192.168.184.27        192.168.1.255        TCP     66 [TCP Retransmission] 61349 → 1688 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1161…  2629.806764  192.168.184.27        192.168.1.255        TCP     66 [TCP Retransmission] 61349 → 1688 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1161…  2635.386084  192.168.184.27        192.168.1.255        TCP     66 [TCP Port numbers reused] 61349 → 1688 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1161…  2650.570982  2409:40e2:16:75ac:a…  64:ff9b::79c7:1638   TCP     86 61350 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1340 WS=256 SACK_PERM
1161…  2651.591274  2409:40e2:16:75ac:a…  64:ff9b::79c7:1638   TCP     86 61351 → 80 [SYN] Seq=0 Win=64320 Len=0 MSS=1340 WS=256 SACK_PERM
1161…  2658.396165  192.168.184.27        192.168.31.97        TCP     66 61352 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1161…  2659.397789  192.168.184.27        192.168.31.97        TCP     66 [TCP Retransmission] 61352 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1161…  2660.486012  192.168.184.27        192.168.1.255        TCP     66 61353 → 1688 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1161…  2661.407629  192.168.184.27        192.168.31.97        TCP     66 [TCP Retransmission] 61352 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1161…  2661.501343  192.168.184.27        192.168.1.255        TCP     66 [TCP Retransmission] 61353 → 1688 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1161…  2663.505734  192.168.184.27        192.168.1.255        TCP     66 [TCP Retransmission] 61353 → 1688 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1161…  2665.417223  192.168.184.27        192.168.31.97        TCP     66 [TCP Retransmission] 61352 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1161…  2667.515851  192.168.184.27        192.168.1.255        TCP     66 [TCP Retransmission] 61353 → 1688 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1161…  2670.980184  192.168.184.27        192.168.31.97        TCP     66 [TCP Port numbers reused] 61352 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
```

```
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 2322008459
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x002 (SYN)
    Window: 65535
    [Calculated window size: 65535]
    Checksum: 0x991d [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP),
  ˅ [Timestamps]
        [Time since first frame in this TCP stream: 0.000000000 sec
        [Time since previous frame in this TCP stream: 0.000000000 :
```

Wireshark · Packet 116156 · Wi-Fi

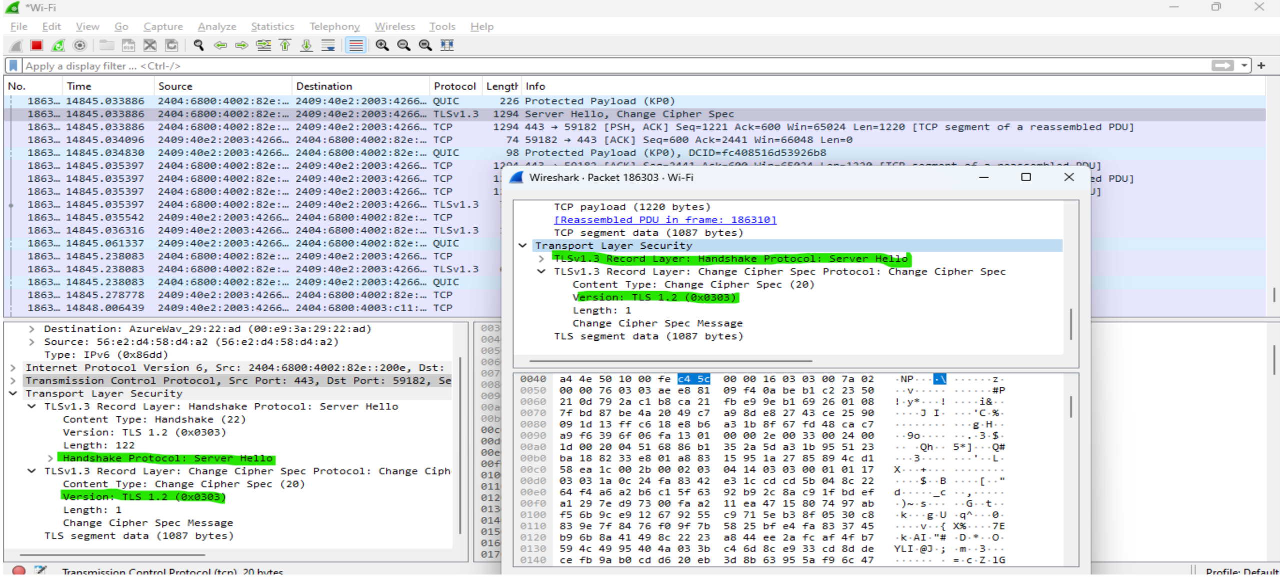
```
  > Flags: 0x002 (SYN)
    Window: 65535
    [Calculated window size: 65535]
    Checksum: 0x991d [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP),
  ˅ [Timestamps]
        [Time since first frame in this TCP stream: 0.000000000 seconds]
        [Time since previous frame in this TCP stream: 0.000000000 seconds]
```

```
0000  56 e2 d4 58 d4 a2 00 e9  3a 29 22 ad 86 dd 60 0d   V··X···· :)"···`·
0010  89 16 00 20 06 fe 24 09  40 e2 00 16 75 ac ac 15   ··· ··$· @···u···
0020  6c 2d f5 4d da 43 00 64  ff 9b 00 00 00 00 00 00   l-·M·C·d ········
0030  00 00 79 c7 16 38 ef a6  00 50 8a 67 09 8b 00 00   ··y··8·· ·P·g····
0040  00 00 80 02 ff ff 99 1d  00 00 02 04 05 3c 01 03   ········ ·····<··
0050  03 08 01 01 04 02                                  ······
```

Figure 23: Understanding TCP flags: Wireshark Packet Details Perspective

Figure 24 displays a Wireshark capture of Transport Layer Security (TLS) traffic, focusing on the "Change Cipher Spec" message, which marks the switch in encryption during a secure session. The packet details pane highlights TLS records and TCP payload data, including hex and ASCII representations of the encrypted segment, illustrating the encryption transition during the handshake process.

Wireshark allows users to analyze TLS handshake packets, particularly the "Client Hello" and "Server Hello" messages that initiate secure communication by negotiating encryption protocols and cipher suites. These handshake details can be inspected through the packet list and details panes, with filters to focus on the TLS handshake exchange.



*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.     Time          Source                   Destination              Protocol  Length Info
1863… 14845.033886  2404:6800:4002:82e:…  2409:40e2:2003:4266…  QUIC       226 Protected Payload (KP0)
1863… 14845.033886  2404:6800:4002:82e:…  2409:40e2:2003:4266…  TLSv1.3   1294 Server Hello, Change Cipher Spec
1863… 14845.033886  2404:6800:4002:82e:…  2409:40e2:2003:4266…  TCP       1294 443 → 59182 [PSH, ACK] Seq=1221 Ack=600 Win=65024 Len=1220 [TCP segment of a reassembled PDU]
1863… 14845.034096  2409:40e2:2003:4266…  2404:6800:4002:82e:…  TCP         74 59182 → 443 [ACK] Seq=600 Ack=2441 Win=66048 Len=0
1863… 14845.034830  2409:40e2:2003:4266…  2404:6800:4002:82e:…  QUIC        98 Protected Payload (KP0), DCID=fc408516d53926b8
1863… 14845.035397  2404:6800:4002:82e:…  2409:40e2:2003:4266…  TCP       1294 443 → 59182 [ACK] Seq=2441 Ack=600 Win=65024 Len=1220 [TCP segment of a reassembled PDU]
1863… 14845.035397  2404:6800:4002:82e:…  2409:40e2:2003:4266…  TCP       1                                                            ed PDU]
1863… 14845.035397  2404:6800:4002:82e:…  2409:40e2:2003:4266…  TCP       1                                                            U]
1863… 14845.035397  2404:6800:4002:82e:…  2409:40e2:2003:4266…  TLSv1.3
1863… 14845.035542  2409:40e2:2003:4266…  2404:6800:4002:82e:…  TCP
1863… 14845.036316  2409:40e2:2003:4266…  2404:6800:4002:82e:…  TLSv1.3
1863… 14845.061337  2409:40e2:2003:4266…  2404:6800:4002:82e:…  QUIC
1863… 14845.238083  2404:6800:4002:82e:…  2409:40e2:2003:4266…  TCP
1863… 14845.238083  2404:6800:4002:82e:…  2409:40e2:2003:4266…  TLSv1.3
1863… 14845.238083  2404:6800:4002:82e:…  2409:40e2:2003:4266…  QUIC
1863… 14845.278778  2409:40e2:2003:4266…  2404:6800:4002:82e:…  TCP
1863… 14848.006439  2409:40e2:2003:4266…  2404:6800:4003:c11:…  TCP
```

```
    > Destination: AzureWav_29:22:ad (00:e9:3a:29:22:ad)
    > Source: 56:e2:d4:58:d4:a2 (56:e2:d4:58:d4:a2)
      Type: IPv6 (0x86dd)
> Internet Protocol Version 6, Src: 2404:6800:4002:82e::200e, Dst:
> Transmission Control Protocol, Src Port: 443, Dst Port: 59182, Se
v Transport Layer Security
    v TLSv1.3 Record Layer: Handshake Protocol: Server Hello
        Content Type: Handshake (22)
        Version: TLS 1.2 (0x0303)
        Length: 122
      > Handshake Protocol: Server Hello
    v TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Ciph
        Content Type: Change Cipher Spec (20)
        Version: TLS 1.2 (0x0303)
        Length: 1
        Change Cipher Spec Message
    TLS segment data (1087 bytes)
```

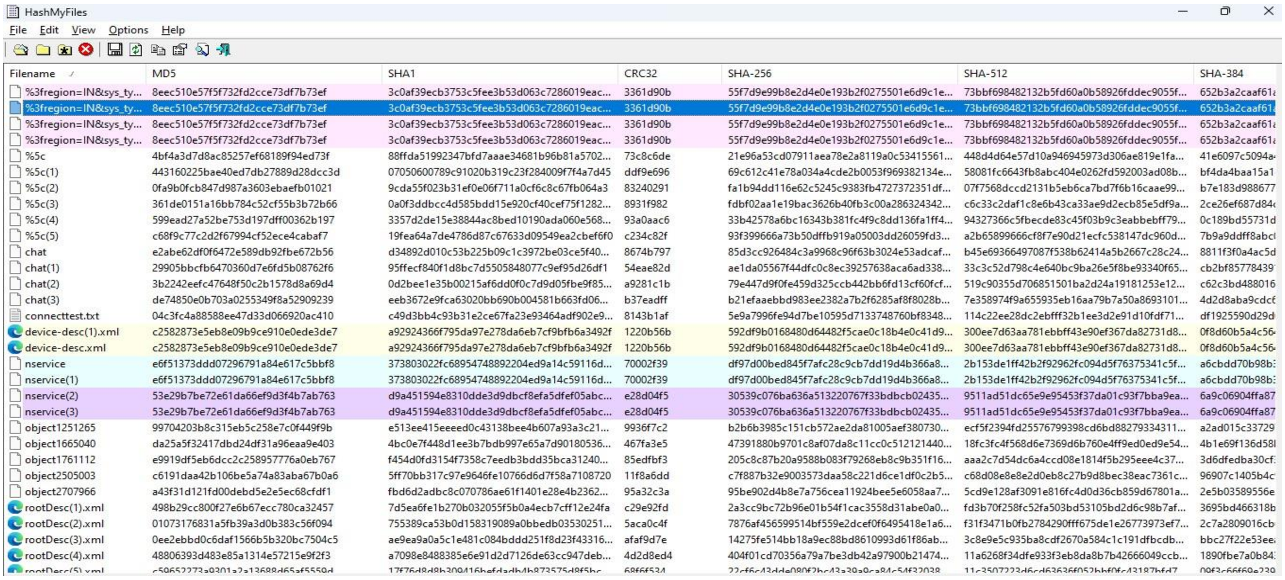
Transmission Control Protocol (tcp), 20 bytes
Profile: Default
Wireshark · Packet 186303 · Wi-Fi

```
      TCP payload (1220 bytes)
      [Reassembled PDU in frame: 186310]
      TCP segment data (1087 bytes)
v Transport Layer Security
    > TLSv1.3 Record Layer: Handshake Protocol: Server Hello
    v TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
        Content Type: Change Cipher Spec (20)
        Version: TLS 1.2 (0x0303)
        Length: 1
        Change Cipher Spec Message
    TLS segment data (1087 bytes)
```

```
0040  a4 4e 50 10 00 fe c4 5c  00 00 16 03 03 00 7a 02   ·NP···\ ······z·
0050  00 00 76 03 03 ae e8 81  09 f4 0a be b1 c2 23 50   ··v····· ······#P
0060  21 0d 79 2a c1 b8 ca 21  fb e9 9e b1 69 26 01 08   !·y*···! ····i&··
0070  7f bd 87 be 4a 20 49 c7  a9 8d e8 27 43 ce 25 90   ····J I· ···'C·%·
0080  09 1d 13 ff c6 18 e8 b6  a3 1b 8f 67 fd 48 ca c7   ········ ···g·H··
0090  a9 f6 39 6f 06 fa 13 01  00 00 2e 00 33 00 24 00   ··9o···· ···.·3·$·
00a0  1d 00 20 04 51 68 86 b1  35 2a 5d a3 1b 95 51 23   ·· ·Qh·· 5*]···Q#
00b0  ba 18 82 33 e8 01 a8 83  15 95 1a 27 85 89 4c d1   ···3···· ···'··L·
00c0  58 ea 1c 00 2b 00 02 03  04 14 03 03 00 01 01 17   X···+··· ········
00d0  03 03 1a 0c 24 fa 83 42  e3 1c cd cd 5b 04 8c 22   ····$··B ····[··"
00e0  64 f4 a6 a2 b6 c1 5f 63  92 b9 2c 8a c9 1f bd ef   d·····_c ··,·····
00f0  a1 29 7e d9 73 00 fa a2  11 ea 47 15 80 74 97 ab   ·)~·s··· ··G··t··
0100  f5 6b 9c e9 12 67 92 55  c9 71 5e b3 8f 05 30 c8   ·k···g·U ·q^···0·
0110  83 9e 7f 84 76 f0 9f 7b  58 25 bf e4 fa 83 37 45   ····v··{ X%····7E
0120  b9 6b 8a 41 49 8c 22 23  a8 44 ee 2a fc af 4f b7   ·k·AI·"# ·D·*··O·
0130  59 4c 49 95 40 4a 03 3b  c4 6d 8c e9 33 cd 8d de   YLI·@J·; ·m··3···
0140  ce fb 9a b0 cd d6 20 eb  3d 8b 63 95 5a f9 6c 47   ······ · =·c·Z·lG
```

Figure 24: Examining TLS Handshake Analysis in Wireshark: Accessing 'Client Hello' and 'Server Hello' Messages

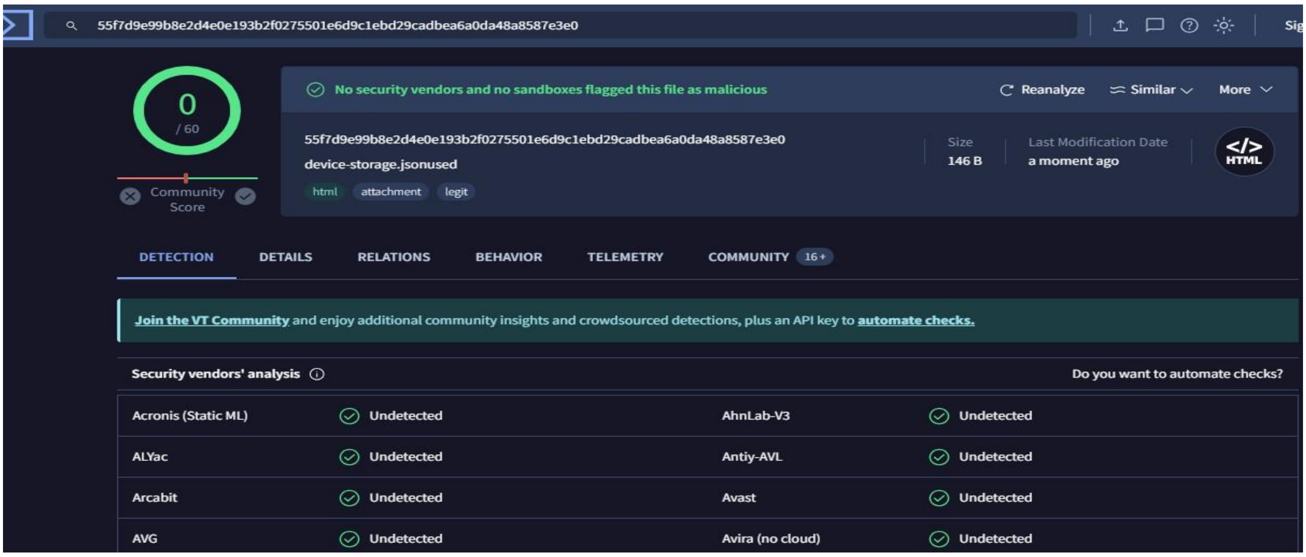
Figure 25 illustrations HashMyFiles, a utility used for calculating cryptographic hash values to ensure file integrity. It displays hash values like MD5, SHA-1, SHA-256, SHA-512, and SHA-384 for each file in a structured table. These algorithms are used for purposes ranging from basic checksum verification to advanced cryptographic security. Highlighted files, such as "Teleponan" and "device-details.xml," may be under investigation for integrity verification, comparison, or auditing. Hashing plays a crucial role in maintaining data authenticity, detecting file tampering, or verifying the integrity of transferred files.



Filename	MD5	SHA1	CRC32	SHA-256	SHA-512	SHA-384
%3fregion=IN&sys_ty...	8eec510e57f5f732fd2cce73df7b73ef	3c0af39ecb3753c5fee3b53d063c7286019eac...	3361d90b	55f7d9e99b8e2d4e0e193b2f0275501e6d9c1e...	73bbf698482132b5fd60a0b58926fddec9055f...	652b3a2caaf61
%3fregion=IN&sys_ty...	8eec510e57f5f732fd2cce73df7b73ef	3c0af39ecb3753c5fee3b53d063c7286019eac...	3361d90b	55f7d9e99b8e2d4e0e193b2f0275501e6d9c1e...	73bbf698482132b5fd60a0b58926fddec9055f...	652b3a2caaf61
%3fregion=IN&sys_ty...	8eec510e57f5f732fd2cce73df7b73ef	3c0af39ecb3753c5fee3b53d063c7286019eac...	3361d90b	55f7d9e99b8e2d4e0e193b2f0275501e6d9c1e...	73bbf698482132b5fd60a0b58926fddec9055f...	652b3a2caaf61
%3fregion=IN&sys_ty...	8eec510e57f5f732fd2cce73df7b73ef	3c0af39ecb3753c5fee3b53d063c7286019eac...	3361d90b	55f7d9e99b8e2d4e0e193b2f0275501e6d9c1e...	73bbf698482132b5fd60a0b58926fddec9055f...	652b3a2caaf61
%5c	4bf4a3d7d8ac85257ef68189f94ed73f	88ffda51992347bfd7aaae34681b96b81a5702...	73c8c6de	21e96a53cd07911aea78e2a8119a0c53415561...	448d4d64e57d10a946945973d306ae819e1fa...	41e6097c5094a
%5c(1)	443160225bae40ed7db27889d28dcc3d	07050600789c91020b319c23f284009f7f4a7d45	ddf9e696	69c612c41e78a034a4cde2b0053f969382134e...	58081fc6643fb8abc404e0262fd592003ad08b...	bf4da4baa15a1
%5c(2)	0fa9b0fcb847d987a3603ebaefb01021	9cda55f023b31ef0e06f711a0cf6c8c67fb064a3	83240291	fa1b94dd116e62c5245c9383fb4727372351df...	07f7568dccd2131b5eb6ca7bd7f6b16caae99...	b7e183d988677
%5c(3)	361de0151a16bb784c52cf55b3b72b66	0a0f3ddbcc4d585bdd15e920cf40cef75f1282...	8931f982	fdbf02aa1e19bac3626b40fb3c00a286324342...	c6c33c2daf1c8e6b43ca33ae9d2ecb85e5df9a...	2ce26ef687d84
%5c(4)	599ead27a52be753d197dff00362b197	3357d2de15e38844ac8bed10190ada060e568...	93a0aac6	33b42578a6bc16343b381fc4f9c8dd136fa1ff4...	94327366c5fbecde83c45f03b9c3eabbebff79...	0c189bd55731d
%5c(5)	c68f9c77c2d2f67994cf52ece4cabaf7	19fea64a7de4786d87c67633d09549ea2cbef6f0	c234c82f	93f399666a73b50dffb919a05003dd26059fd3...	a2b65899666cf8f7e90d21ecfc538147dc960d...	7b9a9ddff8abc
chat	e2abe62df0f6472e589db92fbe672b56	d34892d010c53b225b09c1c3972be03ce5f40...	8674b797	85d3cc926484c3a9968c96f63b3024e53adcaf...	b45e69366497087f538b62414a5b2667c28c24...	8811f3f0a4ac5d
chat(1)	29905bbcfb6470360d7e6fd5b08762f6	95ffecf840f1d8bc7d5505848077c9ef95d26df1	54eae82d	ae1da05567f44dfc0c8ec39257638aca6ad338...	33c3c52d798c4e640bc9ba26e5f8be93340f65...	cb2bf85778439
chat(2)	3b2242eefc47648f50c2b1578d8a69d4	0d2bee1e35b00215af6dd0f0c7d9d05fbe9f85...	a9281c1b	79e447d9f0fe459d325ccb442bb6fd13cf60fcf...	519c90355d706851501ba2d24a19181253e12...	c62c3bd488016
chat(3)	de74850e0b703a0255349f8a52909239	eeb3672e9fca63020bb690b004581b663fd06...	b37eadff	b21efaaebbd983ee2382a7b2f6285af8f8028b...	7e358974f9a655935eb16aa79b7a50a8693101...	4d2d8aba9cdc
connecttest.txt	04c3fc4a88588ee47d33d066920ac410	c49d3bb4c93b31e2ce67fa23e93464adf902e9...	8143b1af	5e9a7996fe94d7be10595d7133748760bf8348...	114c22ee28dc2ebfff32b1ee3d2e91d10fdf71...	df1925590d29d
device-desc(1).xml	c2582873e5eb8e09b9ce910e0ede3de7	a92924366f795da97e278da6eb7cf9bfb6a3492f	1220b56b	592df9b0168480d64482f5cae0c18b4e0c41d9...	300ee7d63aa781ebbff43e90ef367da82731d8...	0f8d60b5a4c56
device-desc.xml	c2582873e5eb8e09b9ce910e0ede3de7	a92924366f795da97e278da6eb7cf9bfb6a3492f	1220b56b	592df9b0168480d64482f5cae0c18b4e0c41d9...	300ee7d63aa781ebbff43e90ef367da82731d8...	0f8d60b5a4c56
nservice	e6f51373ddd07296791a84e617c5bbf8	373803022fc68954748892204ed9a14c59116d...	70002f39	df97d00bed845f7afc28c9cb7dd19d4b366a8...	2b153de1ff42b2f92962fc094d5f76375341c5f...	a6cbdd70b98b
nservice(1)	e6f51373ddd07296791a84e617c5bbf8	373803022fc68954748892204ed9a14c59116d...	70002f39	df97d00bed845f7afc28c9cb7dd19d4b366a8...	2b153de1ff42b2f92962fc094d5f76375341c5f...	a6cbdd70b98b
nservice(2)	53e29b7be72e61da66ef9d3f4b7ab763	d9a451594e8310dde3d9dbcf8efa5dfef05abc...	e28d04f5	30539c076ba636a513220767f33bdbcb02435...	9511ad51dc65e9e95453f37da01c93f7bba9ea...	6a9c06904ffa87
nservice(3)	53e29b7be72e61da66ef9d3f4b7ab763	d9a451594e8310dde3d9dbcf8efa5dfef05abc...	e28d04f5	30539c076ba636a513220767f33bdbcb02435...	9511ad51dc65e9e95453f37da01c93f7bba9ea...	6a9c06904ffa87
object1251265	99704203b8c315eb5c258e7c0f449f9b	e513ee415eeeed0c43138bee4b607a93a3c21...	9936f7c2	b2b6b3985c151cb572ae2da81005aef380730...	ecf5f2394fd25576799398cd6bd88279334311...	a2ad015c33729
object1665040	da25a5f32417dbd24df31a96eaa9e403	4bc0e7f448d1ee3b7bdb997e65a7d90180536...	467fa3e5	47391880b9701c8af07da8c11cc0c512121440...	18fc3fc4f568d6e7369d6b760e4ff9ed0ed9e54...	4b1e69f136d58
object1761112	e9919df5eb6dcc2c258957776a0eb767	f454d0fd3154f7358c7eedb3bdd35bca31240...	85edfbf3	205c8c87b20a9588b083f79268eb8c9b351f16...	aaa2c7d54dc6a4ccd08e1814f5b295eee4c37...	3d6dfedba30cf
object2505003	c6191daa42b106be5a74a83aba67b0a6	5ff70bb317c97e9646fe10766d6d7f58a7108720	11f8a6dd	c7f887b32e9003573daa58c221d6ce1df0c2b5...	c68d08e8e8e2d0eb8c27b9d8bec38eac7361c...	96907c1405b4c
object2707966	a43f31d121fd00debd5e2e5ec68cfdf1	fbd6d2adbc8c070786ae61f1401e28e4b2362...	95a32c3a	95be902d4b8e7a756cea11924bee5e6058aa7...	5cd9e128af3091e816fc4d0d36cb859d67801a...	2e5b03589556e
rootDesc(1).xml	498b29cc800f27e6b67ecc780ca32457	7d5ea6fe1b270b032055f5b0a4ecb7cff12e24fa	c29e92fd	2a3cc9bc72b96e01b54f1cac3558d31abe0a0...	fd3b70f258fc52fa503bd53105bd2d6c98b7af...	3695bd466318b
rootDesc(2).xml	01073176831a5fb39a3d0b383c56f094	755389ca53b0d158319089a0bbedb03530251...	5aca0c4f	7876af456599514bf559e2dcef0f6495418e1a6...	f31f3471b0fb2784290fff675de1e26773973ef7...	2c7a2809016cb
rootDesc(3).xml	0ee2ebbd0c6daf1566b5b320bc7504c5	ae9ea9a0a5c1e481c084bddd251f8d23f43316...	afaf9d7e	14275fe514bb18a9ec88bd8610993d61f86ab...	3c8e9e5c935ba8cdf2670a584c1c191dfbcdb...	bbc27f22e53ee
rootDesc(4).xml	48806393d483e85a1314e57215e9f2f3	a7098e8488385e6e91d2d7126de63cc947deb...	4d2d8ed4	404f01cd70356a79a7be3db42a97900b21474...	11a6268f34dfe933f3eb8da8b7b42666049ccb...	1890fbe7a0b84
rootDesc(5).xml	c59652273a9301a2a13688d65af5559d	17f76d8d8b309416befdadb4b873575d8f5bc	68f6f534	22cf6c43dde080f2bc43a39a9ca84c54f32038	11c3507223d6cd63636f052bbf0fc43187bfd7	09f3c66f69e239

Figure 25: Comprehensive File Integrity and Hash Comparison Using HashMyFiles: A Methodology for Analyzing and Identifying Security Risks Using Wireshark.

Wireshark is employed to analyze malware traffic by identifying and examining network communications associated with malicious software files that initially went undetected or had zero detection by security tools. In this case, the file was not flagged as malicious by any sandboxes or security vendors, as illustrated in Figure 26. Wireshark's capabilities allow for in-depth inspection of such undetected threats.



55f7d9e99b8e2d4e0e193b2f0275501e6d9c1ebd29cadbea6a0da48a8587e3e0

No security vendors and no sandboxes flagged this file as malicious

55f7d9e99b8e2d4e0e193b2f0275501e6d9c1ebd29cadbea6a0da48a8587e3e0
device-storage.jsonused
html attachment legit

Size
146 B

Last Modification Date
a moment ago

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

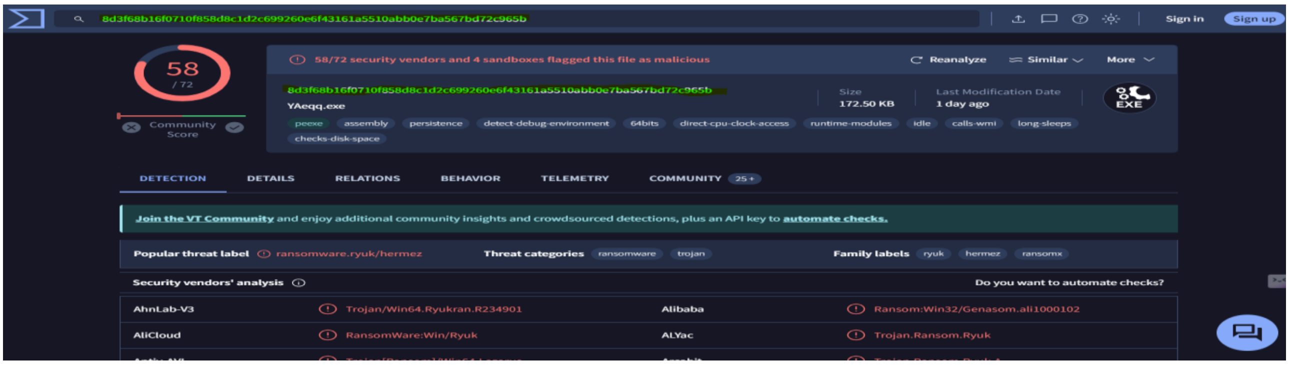
Security vendors' analysis

Do you want to automate checks?

Acronis (Static ML)	Undetected	AhnLab-V3	Undetected
ALYac	Undetected	Antiy-AVL	Undetected
Arcabit	Undetected	Avast	Undetected
AVG	Undetected	Avira (no cloud)	Undetected

Figure 26: Zero Detection: Sandboxes and Security Vendor Results for Non-Malicious File

In this instance, the file was flagged as malicious by various sandboxes and security vendors, as shown in Figure 27. This indicates that the security tools successfully identified the threat, highlighting their effectiveness in detecting potential risks. The detection highlights the importance of using multiple security solutions to enhance the overall protection against malware and other cyber threats. An evaluation of malicious detection results from various sandboxes and security vendors. This assessment demonstrates the effectiveness of these security tools in identifying the threat associated with the analyzed file. The findings highlight the importance of utilizing multiple security solutions to enhance detection capabilities and improve overall protection against malware and other cyber threats in network environments.



8d3f68b16f0710f858d8c1d2c699260e6f43161a5510abb0e7ba567bd72c965b

58/72 security vendors and 4 sandboxes flagged this file as malicious

8d3f68b16f0710f858d8c1d2c699260e6f43161a5510abb0e7ba567bd72c965b
YAeqq.exe
peexe assembly persistence detect-debug-environment 64bits direct-cpu-clock-access runtime-modules idle calls-wmi long-sleeps checks-disk-space

Size
172.50 KB

Last Modification Date
1 day ago

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label ransomware.ryuk/hermez Threat categories ransomware trojan Family labels ryuk hermez ransomx

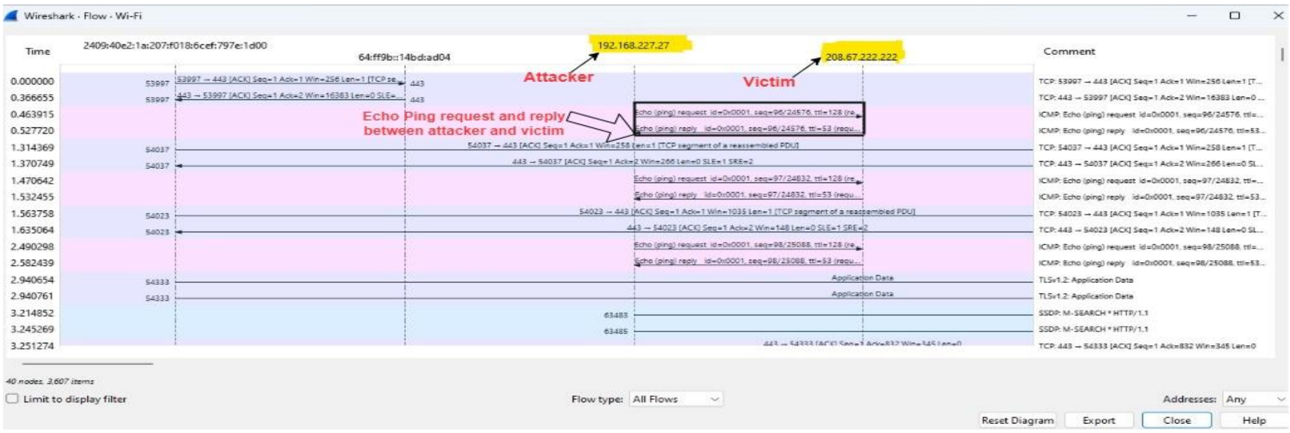
Security vendors' analysis

Do you want to automate checks?

AhnLab-V3	Trojan/Win64.Ryukran.R234901	Alibaba	Ransom:Win32/Genasom.ali1000102
AliCloud	RansomWare:Win/Ryuk	ALYac	Trojan.Ransom.Ryuk

Figure 27: Malicious Detection Evaluation: Sandboxes and Security Vendors' Assessment

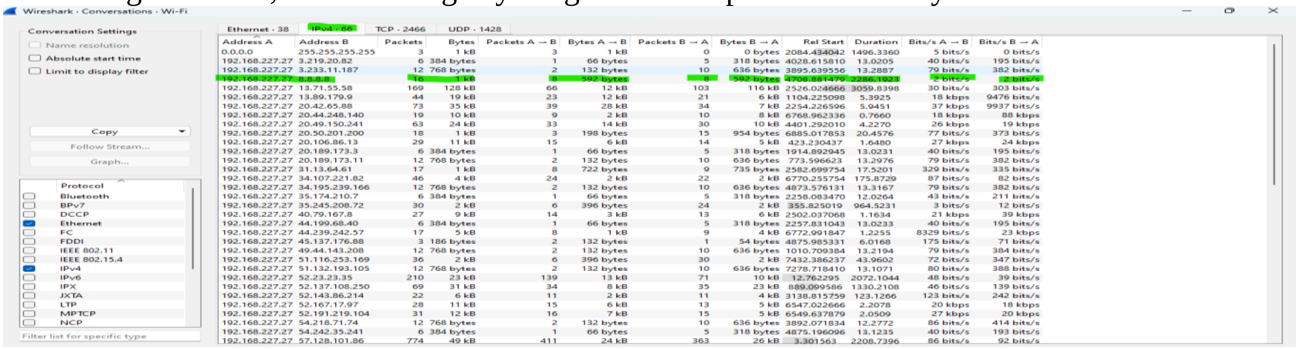
The Flow Graph window in Wireshark is a robust tool for analyzing network traffic, especially in the context of intrusion detection. It provides a visual representation of packet flow between hosts and IP addresses, facilitating the identification of trends and potential security risks, as illustrated in Figure 28. This graphical view enhances the ability to monitor and assess network activity effectively.



Time	Comment
0.000000	TCP: 53997 → 443 [ACK] Seq=1 Ack=1 Win=256 Len=1 [T...
0.366655	TCP: 443 → 53997 [ACK] Seq=1 Ack=2 Win=16383 Len=0 ...
0.463915	ICMP: Echo (ping) request id=0x0001, seq=96/24576, ttl=...
0.527720	ICMP: Echo (ping) reply id=0x0001, seq=96/24576, ttl=53...
1.314369	TCP: 54037 → 443 [ACK] Seq=1 Ack=1 Win=258 Len=1 [T...
1.370749	TCP: 443 → 54037 [ACK] Seq=1 Ack=2 Win=266 Len=0 SL...
1.470642	ICMP: Echo (ping) request id=0x0001, seq=97/24832, ttl=...
1.532455	ICMP: Echo (ping) reply id=0x0001, seq=97/24832, ttl=53...
1.563758	TCP: 54023 → 443 [ACK] Seq=1 Ack=1 Win=1035 Len=1 [T...
1.635064	TCP: 443 → 54023 [ACK] Seq=1 Ack=2 Win=148 Len=0 SL...
2.490298	ICMP: Echo (ping) request id=0x0001, seq=98/25088, ttl=...
2.582439	ICMP: Echo (ping) reply id=0x0001, seq=98/25088, ttl=53...
2.940654	TLSv1.2: Application Data
2.940761	TLSv1.2: Application Data
3.214852	SSDP: M-SEARCH * HTTP/1.1
3.245269	SSDP: M-SEARCH * HTTP/1.1
3.251274	TCP: 443 → 54333 [ACK] Seq=1 Ack=832 Win=345 Len=0

Figure 28: The Flow Graph window for Intrusion Detection

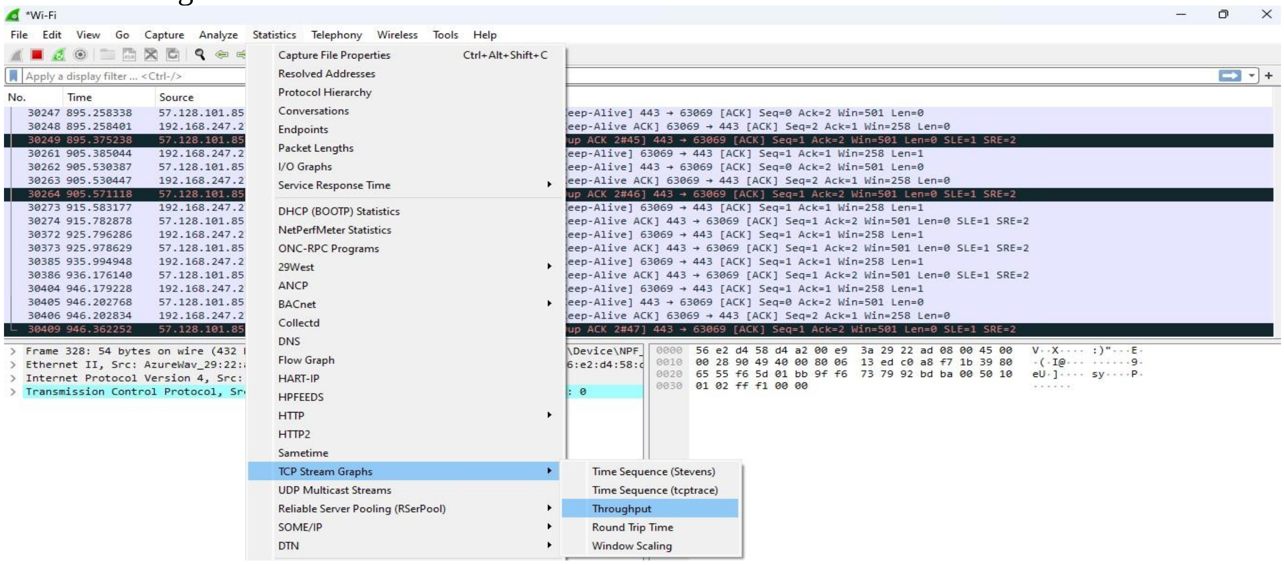
Figure 29 illustrates the Conversation section in Wireshark, which shows the entities involved in the network communication. It provides detailed information about the number of packets and bytes exchanged between the communicating parties. This feature helps in identifying traffic patterns, monitoring data flow, and detecting any irregularities or potential security threats within the network.



Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
0.0.0.0	255.255.255.255	3	1 kB	3	1 kB	0	0 bytes	2084.434042	1496.3360	5 bits/s	0 bits/s
192.168.227.27	3.219.20.82	6	384 bytes	1	66 bytes	5	318 bytes	4028.615810	13.0205	40 bits/s	195 bits/s
192.168.227.27	3.233.11.187	12	768 bytes	2	132 bytes	10	636 bytes	3895.639556	13.2887	79 bits/s	382 bits/s
192.168.227.27	8.8.8.8	16	1 kB	8	592 bytes	8	592 bytes	4708.881479	2286.1923	2 bits/s	2 bits/s
192.168.227.27	13.71.55.58	169	128 kB	66	12 kB	103	116 kB	2526.024666	3059.8398	30 bits/s	303 bits/s
192.168.227.27	13.89.179.9	44	19 kB	23	12 kB	21	6 kB	1104.225098	5.3925	18 kbps	9476 bits/s
192.168.227.27	20.42.65.88	73	35 kB	39	28 kB	34	7 kB	2254.226596	5.9451	37 kbps	9937 bits/s
192.168.227.27	20.44.248.140	19	10 kB	9	2 kB	10	8 kB	6768.962336	0.7660	18 kbps	88 kbps
192.168.227.27	20.49.150.241	63	24 kB	33	14 kB	30	10 kB	4401.292010	4.2270	26 kbps	19 kbps
192.168.227.27	20.50.201.200	18	1 kB	3	198 bytes	15	954 bytes	6885.017853	20.4576	77 bits/s	373 bits/s
192.168.227.27	20.106.86.13	29	11 kB	15	6 kB	14	5 kB	423.230437	1.6480	27 kbps	24 kbps
192.168.227.27	20.189.173.3	6	384 bytes	1	66 bytes	5	318 bytes	1914.892945	13.0231	40 bits/s	195 bits/s
192.168.227.27	20.189.173.11	12	768 bytes	2	132 bytes	10	636 bytes	773.596623	13.2976	79 bits/s	382 bits/s
192.168.227.27	31.13.64.61	17	1 kB	8	722 bytes	9	735 bytes	2582.699754	17.5201	329 bits/s	335 bits/s
192.168.227.27	34.107.221.82	46	4 kB	24	2 kB	22	2 kB	6770.255754	175.8729	87 bits/s	82 bits/s
192.168.227.27	34.195.239.166	12	768 bytes	2	132 bytes	10	636 bytes	4873.576131	13.3167	79 bits/s	382 bits/s
192.168.227.27	35.174.210.7	6	384 bytes	1	66 bytes	5	318 bytes	2258.083470	12.0264	43 bits/s	211 bits/s
192.168.227.27	35.245.208.72	30	2 kB	6	396 bytes	24	2 kB	355.825019	964.5231	3 bits/s	12 bits/s
192.168.227.27	40.79.167.8	27	9 kB	14	3 kB	13	6 kB	2502.037068	1.1634	21 kbps	39 kbps
192.168.227.27	44.199.68.40	6	384 bytes	1	66 bytes	5	318 bytes	2257.831043	13.0233	40 bits/s	195 bits/s
192.168.227.27	44.239.242.57	17	5 kB	8	1 kB	9	4 kB	6772.991847	1.2255	8329 bits/s	23 kbps
192.168.227.27	45.137.176.88	3	186 bytes	2	132 bytes	1	54 bytes	4875.985331	6.0168	175 bits/s	71 bits/s
192.168.227.27	49.44.143.208	12	768 bytes	2	132 bytes	10	636 bytes	1010.709384	13.2194	79 bits/s	384 bits/s
192.168.227.27	51.116.253.169	36	2 kB	6	396 bytes	30	2 kB	7432.386237	43.9602	72 bits/s	347 bits/s
192.168.227.27	51.132.193.105	12	768 bytes	2	132 bytes	10	636 bytes	7278.718410	13.1071	80 bits/s	388 bits/s
192.168.227.27	52.23.23.35	210	23 kB	139	13 kB	71	10 kB	12.762295	2072.1044	48 bits/s	39 bits/s
192.168.227.27	52.137.108.250	69	31 kB	34	8 kB	35	23 kB	889.099586	1330.2108	46 bits/s	139 bits/s
192.168.227.27	52.143.86.214	22	6 kB	11	2 kB	11	4 kB	3138.815759	123.1266	123 bits/s	242 bits/s
192.168.227.27	52.167.17.97	28	11 kB	15	6 kB	13	5 kB	6547.022666	2.2078	20 kbps	18 kbps
192.168.227.27	52.191.219.104	31	12 kB	16	7 kB	15	5 kB	6549.637879	2.0509	27 kbps	20 kbps
192.168.227.27	54.218.71.74	12	768 bytes	2	132 bytes	10	636 bytes	3892.071834	12.2772	86 bits/s	414 bits/s
192.168.227.27	54.242.35.241	6	384 bytes	1	66 bytes	5	318 bytes	4875.196096	13.1235	40 bits/s	193 bits/s
192.168.227.27	57.128.101.86	774	49 kB	411	24 kB	363	26 kB	3.301563	2208.7396	86 bits/s	92 bits/s

Figure 29: The conversation section of IPV4 packets for byte exchange

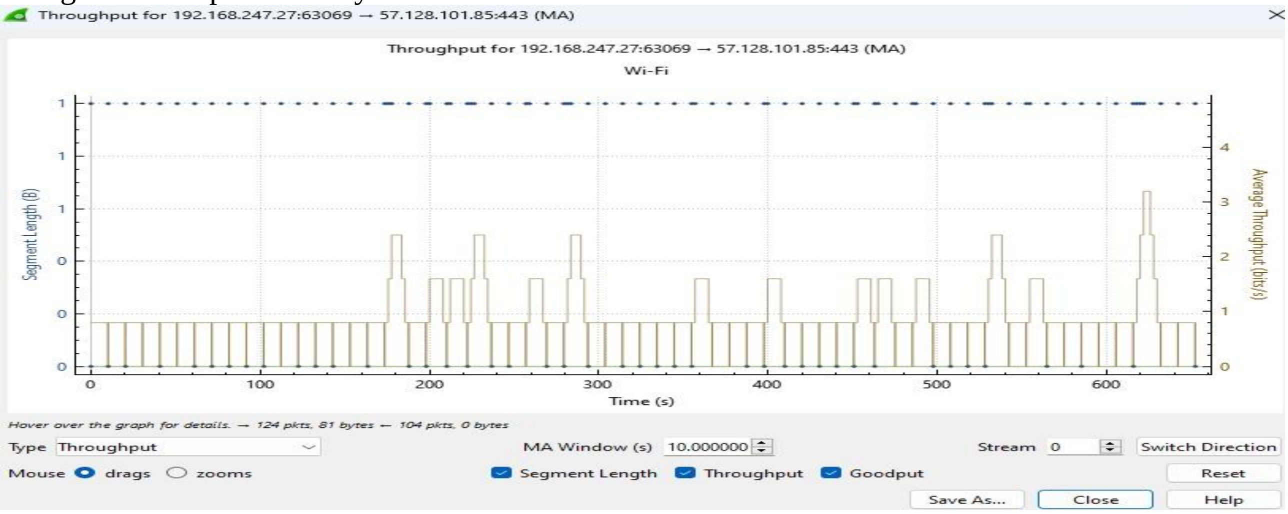
Figure 30 depicts the TCP Stream Throughput graph, which visualizes the throughput of a single TCP stream in one direction, calculated from the selected packet. This graph provides critical insights into the data transfer rate over time, helping to assess network performance, detect congestion, and identify any abnormalities in the TCP stream's behavior, essential for optimizing traffic analysis and troubleshooting.



```
No.   Time        Source
30247 895.258338  57.128.101.85     eep-Alive] 443 → 63069 [ACK] Seq=0 Ack=2 Win=501 Len=0
30248 895.258401  192.168.247.2     eep-Alive ACK] 63069 → 443 [ACK] Seq=2 Ack=1 Win=258 Len=0
30249 895.375238  57.128.101.85     up ACK 2#45] 443 → 63069 [ACK] Seq=1 Ack=2 Win=501 Len=0 SLE=1 SRE=2
30261 905.385044  192.168.247.2     eep-Alive] 63069 → 443 [ACK] Seq=1 Ack=1 Win=258 Len=1
30262 905.530387  57.128.101.85     eep-Alive] 443 → 63069 [ACK] Seq=0 Ack=2 Win=501 Len=0
30263 905.530447  192.168.247.2     eep-Alive ACK] 63069 → 443 [ACK] Seq=2 Ack=1 Win=258 Len=0
30264 905.571118  57.128.101.85     up ACK 2#46] 443 → 63069 [ACK] Seq=1 Ack=2 Win=501 Len=0 SLE=1 SRE=2
30273 915.583177  192.168.247.2     eep-Alive] 63069 → 443 [ACK] Seq=1 Ack=1 Win=258 Len=1
30274 915.782878  57.128.101.85     eep-Alive ACK] 443 → 63069 [ACK] Seq=1 Ack=2 Win=501 Len=0 SLE=1 SRE=2
30372 925.796286  192.168.247.2     eep-Alive] 63069 → 443 [ACK] Seq=1 Ack=1 Win=258 Len=1
30373 925.978629  57.128.101.85     eep-Alive ACK] 443 → 63069 [ACK] Seq=1 Ack=2 Win=501 Len=0 SLE=1 SRE=2
30385 935.994948  192.168.247.2     eep-Alive] 63069 → 443 [ACK] Seq=1 Ack=1 Win=258 Len=1
30386 936.176140  57.128.101.85     eep-Alive ACK] 443 → 63069 [ACK] Seq=1 Ack=2 Win=501 Len=0 SLE=1 SRE=2
30404 946.179228  192.168.247.2     eep-Alive] 63069 → 443 [ACK] Seq=1 Ack=1 Win=258 Len=1
30405 946.202768  57.128.101.85     eep-Alive] 443 → 63069 [ACK] Seq=0 Ack=2 Win=501 Len=0
30406 946.202834  192.168.247.2     eep-Alive ACK] 63069 → 443 [ACK] Seq=2 Ack=1 Win=258 Len=0
30409 946.362252  57.128.101.85     up ACK 2#47] 443 → 63069 [ACK] Seq=1 Ack=2 Win=501 Len=0 SLE=1 SRE=2

> Frame 328: 54 bytes on wire (432     \Device\NPF_
> Ethernet II, Src: AzureWav_29:22:    6:e2:d4:58:d
> Internet Protocol Version 4, Src:
> Transmission Control Protocol, Sr    : 0

0000  56 e2 d4 58 d4 a2 00 e9  3a 29 22 ad 08 00 45 00   V··X···· :)"···E·
0010  00 28 90 49 40 00 80 06  13 ed c0 a8 f7 1b 39 80   ·(·I@··· ······9·
0020  65 55 f6 5d 01 bb 9f f6  73 79 92 bd ba 00 50 10   eU·]···· sy····P·
0030  01 02 ff f1 00 00                                  ······
```

Figure 30: TCP Stream Graph and Throughput Window for Intrusion Detection

Figure 31 illustrates Wireshark's TCP Stream Graph and Throughput Graph, tools that provide detailed visual analysis of TCP communication and network performance. These graphs enable users to evaluate data transmission efficiency, monitor packet flow, detect bottlenecks, and identify retransmissions or delays. Such insights are critical for diagnosing network issues and optimizing TCP connections in real-time. The TCP Stream Graph shows the progression of packets between sender and receiver, revealing delays or retransmissions. The Throughput Graph highlights data transfer rate fluctuations, aiding in diagnosing congestion and performance drops. Wireshark offers network engineers real-time tools to assess traffic, optimize bandwidth, and improve overall performance through detailed packet analysis.


Throughput for 192.168.247.27:63069 → 57.128.101.85:443 (MA)

Throughput for 192.168.247.27:63069 → 57.128.101.85:443 (MA)
Wi-Fi
Segment Length (B)
1
1
1
0
0
0
Average Throughput (bits/s)
4
3
2
1
0
0
100
200
300
400
500
600
Time (s)

Hover over the graph for details. → 124 pkts, 81 bytes ← 104 pkts, 0 bytes

Type Throughput

MA Window (s) 10.000000

Stream 0

Mouse ● drags ○ zooms

☑ Segment Length ☑ Throughput ☑ Goodput

Figure 31: TCP StreamGraph and Throughput Graph

6. Results and Discussion

Wireshark captures real-time data packets from the network, storing them in .pcap files for detailed analysis. Along with Snort, an open-source intrusion detection system compatible with both Windows and Linux, Wireshark is widely used for developing intrusion detection systems. It provides a comprehensive visual representation of network elements, including IP addresses, protocols, timestamps, header lengths, and service categories, with payload data in hexadecimal format and header data in decimal format. The I/O graph in Wireshark plots the number of packets captured per second on the y-axis against time on the x-axis. For example, ping requests are sent to the victim's IP address (208.67.222.222) from the attacker's IP address (192.168.128.27), as shown in Fig. 13. The flow graph visually represents network traffic, including IP addresses, port numbers, and protocols, aiding in network troubleshooting and traffic analysis. Wireshark's expert information flags issues like malformed packets, marked in red according to its coloring rules (e.g., bad TCP packets in Fig. 20), which raises suspicions. Additionally, Wireshark helps analyze malware traffic by identifying network communications linked to malicious software, even when undetected by sandboxes or security vendors, as shown in Figs. 26 and 27. Wireshark also supports network file integrity analysis through external hashing tools, as depicted in Fig. 25. For connection stability, the Throughput Graph window of the TCP stream graphs helps evaluate throughput, as shown in Fig. 31.

Our evaluation reveals that SNORT and WIRESHARK are effective intrusion detection tools in modern network environments, but continuous improvements and integrations are crucial for maintaining their efficacy in protecting critical assets and data. Enterprises can enhance their network monitoring and defense against online threats by adopting these tools.

7. Conclusion and Future Scope

Wireshark is a powerful tool for detecting network intrusions, offering advanced features such as display filters, I/O graphs, color-coded packet identification, and detailed traffic analysis. In today's digital landscape, network security is critical, and proactive monitoring is essential for identifying and preventing cyber threats. Integrating Wireshark with Snort, a leading Intrusion Detection System

(IDS), allows for efficient traffic analysis and enhanced network protection. This paper examines Snort's IDS, which uses a rule-based detection engine to identify malicious traffic patterns through tools like the flow graph window. Snort logs suspicious data packets and alert messages, which are then exported to Wireshark for in-depth packet-level analysis. This integration enables effective correlation of events and detection of security breaches. Our evaluation highlights the effectiveness of Snort and Wireshark as IDS tools, emphasizing their combined ability to detect both known vulnerabilities and emerging threats. By leveraging Snort's signature-based detection with Wireshark's packet inspection capabilities, these tools provide comprehensive defense mechanisms to safeguard network infrastructures against cyberattacks.

Wireshark, when utilized alongside Snort, Syslog, and other intrusion detection mechanisms, cannot independently manage network intrusions. This research focuses on capturing real-time network traffic and analyzing packets using Wireshark and Snort. While Wireshark excels in traffic analysis, it lacks the capability to alert users or take corrective actions against unauthorized access attempts. Intrusion Detection Systems (IDS) are critical for identifying suspicious activities and mitigating potential risks. Current signature-based IDS solutions often face performance limitations, such as slow processing speeds and high memory usage. In my forthcoming research, we will compare various traffic analysis tools, emphasizing Wireshark's advantages as a network protocol analyzer across multiple domains. We propose enhancing its functionality by integrating additional utilities into the source code to improve alert generation and heuristic detection. Evaluating Snort and Wireshark as IDS tools provides a solid foundation for advancing network security. While both tools are vital for analyzing network traffic and identifying cyber threats, their effectiveness hinges on continuous innovation to address evolving attack vectors. The future of Snort and Wireshark as IDS solutions depends on their ability to enhance real-time detection capabilities and deliver comprehensive network traffic visibility, all while tackling challenges related to scalability and usability. Ongoing research and development are essential to ensure these tools remain effective in safeguarding network infrastructures against malicious activities.

Acknowledgements

I would like to express my sincere gratitude to my mentor, Dr. Bichitrananda Behera, from the Department of Computer Science and Engineering at C.V. Raman Global University, Bhubaneswar, Odisha, for his invaluable guidance and support throughout the completion of this research. His encouragement and insights were instrumental in shaping the direction of my work. I would also like to extend my heartfelt thanks to my family and brother for their unwavering advice and constant support. I am honored to have had the opportunity to pursue my research with their help.

Conflict of Interest

The authors declare that they have no conflict of interest.

Competing Interests

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Tripathy Sudhanshu Sekhar, & Behera Bichitrananda. (2023). PERFORMANCE EVALUATION OF MACHINE LEARNING ALGORITHMS FOR INTRUSION DETECTION SYSTEM. Journal of biomechanical science and engineering, April theme-ii, 621–640.

[2] Singh, & Kumar. (2020, March 2). Network Forensics using Snort and Wireshark. International Journal of Future Generation Communication and Networking, 13(3), 509–515.

[3] Banerjee, Usha & Vashishtha, Ashutosh & Mukul, Saxena. (2010). Evaluation of the Capabilities of WireShark as a Tool for Intrusion Detection. International Journal of Computer Applications.

[4] Hebbar, R., & Mohan, K. (2015). Packet analysis with Network Intrusion Detection System. Int. J. Sci. Res. IJSR, 4(2).

[5] Vivens Ndatinya, Zhifeng Xiao, Ke Meng, “Network Forensic Analysis using Wireshark”, International Journal of Sensor Networks, vol. 10, Issue No. 2, 2015.

[6] Saboor, M. Akhlaq, B. Aslam, “Experimental Evaluation of Snort against DDOS Attacks under different Hardware Configurations”, 2nd National Conference of Information Assurance, (2013).

[7] Qing-Xiu Wu, “The Network Protocol Analysis Technique in Snort”, International Conference on Solid State Devices and Materials Science, (2012).

[8] Vijayarani S. and Sylviaa S. (2015). Intrusion detection system– a study. International Journal of Security Privacy and Trust Management, 4(1), 31-44.

[9] Pavithirakini, S., Bandara, S., Gunawardhana, C., Perera, K.K., Abeyrathne, B.G., & Dhammearatchi, D. (2016). Improve the Capabilities of Wireshark as a tool for Intrusion Detection in DOS Attacks.

[10] Pramudya, P. B., & Alamsyah, A. (2022). Implementation of a signature-based intrusion detection system using SNORT to prevent threats in network servers. Journal of Soft Computing Exploration, 3(2), 93 – 98.

[11] Nitin Verma, “Detect Network Threat Using SNORT Intrusion Detection System,” International Research Journal of Engineering and Technology (IRJET), vol. 09, no. 01, Jan. 2022.

[12] Singh, Gopal and Goyal, Sachin and Agarwal, Ratish, Intrusion Detection Using Network Monitoring Tools (April 17, 2014).

[13] T. Lappas, and K. Pelechrinis "Data Mining Techniques for (Network) Intrusion Detection Systems", Department of Computer Science and Engineering, UC Riverside, Riverside CA 92521.

[14] J. Gomez, C. Gil, N. Padilla, R. Banos and C. Jimenez, "Design of a Snort-Based Hybrid Intrusion Detection System". Omatu et al. (Eds.): IWANN (The International Work Conference on Artificial Neural Networks) 2009, Part II, LNCS 5518, 2009, pp. 515–522.

[15] A. S. Ashoor and S. Gore "Intrusion Detection System (IDS) & Intrusion Prevention System (IPS): Case Study", International Journal of Scientific & Engineering Research, Volume 2, Issue 7, July-2011, ISSN 2229-5518.

[16] R. S. Shirbhate and P. A. Patil "Network Traffic Monitoring Using Intrusion Detection System", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 1, January 2012, ISSN: 2277 128X.

[17] T. Holland, "Understanding IPS and IDS: Using IPS and IDS together for Defense in Depth" by GSEC Practical v1.4b, Option 1, SANS Institute February 23, 2004.

[18] A. Youssef and A. Emam, "Network Intrusion Detection Using Data Mining and Network Behavior.

[19] Analysis", International Journal of Computer Science & Information Technology (IJCSIT) Vol. 3, No 6, Dec 2011, Page No. 87-98.

[20] S. Gupta and R. Mamtora, "Intrusion Detection System Using Wireshark", International Journal of Advanced Research in Computer Science and Software Engineering, ISSN: 2277 128X, Volume 2, Issue 11, November 2012, Page No. 358-363.

[21] Abhinav Singh, "Instant Wireshark Starter" Packt Publishing Ltd., So, what is Wireshark? Page no 3, eBook.

[22] What is Snort? Andrew R. Baker, Brian Caswell and Mike Poor "Snort 2.1, Intrusion Detection", Syngress Publishing, Inc, eBook page no. 55-56.