Real-Time Network Traffic Monitoring System

PROJECT REPORT

Submitted in the partial fulfilment of the requirements for award of the

Six Months Online Certificate Course

in

CYBER SECURITY

Course Duration: [25-01-2024 to 24-07-2024]

By

Habeeb ur Rahman MD (Ht. No.2406CYS120)

Under the Esteemed Guidance Prof. P PENCHALAIAH

DIRECTORATE OF INNOVATIVE LEARNING & TEACHING (DILT)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

(Formerly SCDE_SCHOOL OF CONTINUING AND DISTANCE EDUCATION)

Kukatpally, Hyderabad, Telangana State, INDIA- 500 085

JULY 2024

ABSTRACT

The Real-Time Network Traffic Monitoring System (RTNTMS) is a comprehensive solution designed to provide real-time insights into network traffic patterns and anomalies. In today's interconnected world, where network security and performance are paramount, the ability to monitor and analyze network traffic in real-time is crucial. RTNTMS employs advanced data collection techniques and analysis algorithms to monitor incoming and outgoing network packets, identify potential security threats, detect abnormal behavior, and optimize network performance. The system offers a user-friendly interface for administrators to visualize network traffic data, set alerts for suspicious activities, and take proactive measures to safeguard the network. By leveraging the power of real-time analytics, RTNTMS empowers organizations to enhance their network security posture, improve operational efficiency, and ensure a seamless user experience. This project serves as a valuable tool for network administrators, cybersecurity professionals, and organizations seeking to fortify their network infrastructure in today's dynamic digital landscape.

TABLE OF CONTENTS

CHAPTER1

Introduction

1. History of Networking

1.1 Early Beginnings

The concept of networking dates back to the 1960s with the inception of ARPANET (Advanced Research Projects Agency Network). Funded by the U.S. Department of Defense, ARPANET aimed to connect computers across geographically separated locations, facilitating communication and resource sharing. ARPANET used packet switching, a method of grouping data into packets for transmission, which laid the groundwork for modern network protocols.

1.2 Development of Ethernet and TCP/IP

In the 1970s and 1980s, Ethernet and TCP/IP emerged as critical technologies. Ethernet, developed by Xerox PARC, provided a standard for connecting devices in a local area network (LAN). It enabled high-speed data transfer over coaxial cables and later twisted pair and fiber optics. TCP/IP, a suite of communication protocols, was developed to standardize data exchange over interconnected networks, forming the backbone of the internet.

1.3 The World Wide Web

The early 1990s saw the development of the World Wide Web by Tim Berners-Lee. This innovation transformed the internet from a primarily academic and government tool into a global information-sharing platform accessible to the general public. The introduction of web browsers and HTML (Hypertext Markup Language) enabled the creation of user-friendly websites, accelerating the internet's growth and utility.

1.4 Modern Networking

Today, networking encompasses a wide range of technologies, including wireless communication, broadband internet, and cloud computing. Wireless networking, enabled by Wi-Fi and cellular technologies, allows devices to connect without physical cables, providing mobility and convenience. Broadband internet offers high-speed data transmission, supporting bandwidth-intensive applications such as streaming and online gaming. Cloud computing leverages remote servers to store, manage, and process data, enabling scalable and flexible IT resource.

2. Importance of Networking

2.1 Facilitating Communication

Networking enables seamless communication between individuals and organizations, supporting various forms of interaction such as emails, instant messaging, voice calls, and video conferencing.

This connectivity fosters collaboration, enhances productivity, and enables real-time decision-making.

2.2 Resource Sharing

Networks allow multiple devices to share resources such as files, printers, and internet connections. This shared access reduces costs, improves efficiency, and simplifies management by centralizing resources. For example, a single printer can serve multiple computers in an office, eliminating the need for individual printers for each device.

2.3 Data Exchange and Collaboration

Networking facilitates the transfer of data between devices, essential for business operations, scientific research, and personal use. It supports collaborative work by allowing multiple users to access and work on the same data simultaneously. For instance, cloud-based collaboration tools enable teams to edit documents, share files, and communicate in real-time, regardless of their physical location.

2.4 Remote Access

Networking allows remote access to systems and data, enabling telecommuting and remote administration. Employees can work from anywhere with internet access, increasing flexibility and reducing the need for physical office space. Remote administration tools enable IT professionals to manage and troubleshoot systems without being physically present, enhancing operational efficiency.

2.5 Enhancing Security

Properly configured networks provide security features such as firewalls, encryption, and access controls to protect data from unauthorized access. Firewalls monitor and control incoming and outgoing network traffic based on security rules, while encryption scrambles data to make it unreadable to unauthorized users. Access controls restrict network access to authorized users, ensuring data confidentiality and integrity.

3. Possible Cyber Attacks on Networks

3.1 Denial of Service (DoS) Attacks

DoS attacks aim to overwhelm a network or server with excessive traffic, rendering it unavailable to legitimate users. Attackers flood the target with a massive volume of requests, exhausting its

resources and causing service disruptions. Distributed Denial of Service (DDoS) attacks involve multiple compromised systems, amplifying the attack's impact.

3.2 Man-in-the-Middle (MitM) Attacks

In MitM attacks, attackers intercept and potentially alter communication between two parties without their knowledge. By positioning themselves between the victim and the intended recipient, attackers can eavesdrop on conversations, steal sensitive information, or inject malicious data. These attacks exploit weaknesses in network protocols and encryption.

3.3 Phishing

Phishing attacks involve tricking users into revealing sensitive information, such as login credentials or financial details, by posing as a legitimate entity. Attackers use emails, websites, or instant messages to lure victims into providing confidential data. Spear phishing, a targeted variant, focuses on specific individuals or organizations, often using personalized information to increase credibility.

3.4 Malware

Malware is malicious software designed to infiltrate and damage computer systems or steal data. Common types of malwares include viruses, worms, ransomware, and spyware. Viruses attach to legitimate programs and spread when the infected program is executed, while worms replicate themselves and spread across networks. Ransomware encrypts data and demands payment for its release, and spyware monitors user activity to gather sensitive information.

3.5 SQL Injection

SQL injection attacks exploit vulnerabilities in web applications to execute malicious SQL queries, accessing or modifying data in a database. Attackers manipulate input fields, such as login forms or search boxes, to inject malicious SQL code, bypassing authentication and gaining unauthorized access to sensitive data.

3.6 Eavesdropping

Eavesdropping involves unauthorized interception of data transmitted over a network. Attackers use packet sniffing techniques to capture and analyze network traffic, potentially gaining access

to sensitive information such as login credentials, financial data, or personal communications. Encryption and secure communication protocols can mitigate the risk of eavesdropping.

4. What is Packet Sniffing?

4.1 Definition and Purpose

Packet sniffing is the process of capturing, analyzing, and interpreting network packets as they traverse a network. Network packets are small units of data transmitted over a network, containing information such as source and destination addresses, protocol type, and payload. Packet sniffing allows network administrators to monitor and analyze network traffic, identify issues, and optimize performance.

4.2 How Packet Sniffing Works

Packet sniffing involves intercepting network packets and examining their contents. This can be done using hardware devices or software tools. The captured packets are stored for analysis, providing insights into network behavior and performance. Packet sniffers operate in promiscuous mode, allowing them to capture all packets on a network segment, regardless of the intended recipient.

4.3 Applications of Packet Sniffing

Packet sniffing has various applications, including network troubleshooting, performance analysis, and security monitoring. It helps network administrators identify and resolve issues such as network congestion, protocol errors, and configuration problems. In security monitoring, packet sniffing is used to detect potential threats, such as unauthorized access, malware infections, and data exfiltration.

5. Packet Sniffing

5.1 Early Packet Sniffing Tools

The concept of packet sniffing dates back to the early days of networking. In the 1980s, as networks became more complex, there was a growing need for tools to monitor and analyze network traffic. Early packet sniffers were rudimentary and often required specialized hardware. These tools were primarily used by network engineers and researchers to understand network behavior and diagnose issues.

5.2 Development of Tcpdump

In the late 1980s, Tcpdump was developed by Van Jacobson, Craig Leres, and Steven McCanne at the Lawrence Berkeley National Laboratory. Tcpdump is a command-line packet analyzer that captures and displays network packets in real-time. It became one of the first widely-used packet

sniffing tools, providing a powerful yet accessible means of analyzing network traffic. Tcpdump's open-source nature allowed for continuous development and improvement by the networking community.

5.3 Emergence of Wireshark

In the 2000s, Gerald Combs developed Ethereal, which was later renamed Wireshark. Wireshark revolutionized packet sniffing by providing a graphical user interface (GUI) that simplified the process of capturing and analyzing network packets. Its extensive protocol support, advanced filtering capabilities, and user-friendly interface made it the tool of choice for network administrators, security analysts, and researchers. Wireshark's open-source model ensured continuous enhancements and updates, keeping it at the forefront of network analysis tools.

6. Types of Packet Sniffing

6.1 Passive Sniffing

Passive sniffing involves capturing packets from a network without altering the data or interfering with network operations. It is typically used in environments where data is transmitted over a shared medium, such as Ethernet hubs.

6.1.1 How Passive Sniffing Works

In passive sniffing, the sniffer listens to all traffic passing through the network and captures packets for analysis. The captured packets are stored and analyzed to understand network behavior, identify issues, and optimize performance.

6.1.2 Advantages of Passive Sniffing

Non-Intrusive: Passive sniffing does not affect network performance or alter data.

Stealthy: It is difficult for attackers to detect passive sniffing, making it suitable for monitoring and security purposes.

Comprehensive: Passive sniffing can capture all traffic on a shared medium, providing a complete view of network activity.

6.1.3 Disadvantages of Passive Sniffing

Limited to Shared Mediums: Passive sniffing is effective only on networks with shared mediums, such as hubs or broadcast domains.

Switched Networks: On switched networks, passive sniffing cannot capture packets not destined for the sniffer's network interface without additional techniques.

6.2 Active Sniffing

Active sniffing involves actively interacting with the network to capture packets. This can include techniques such as Address Resolution Protocol (ARP) spoofing, MAC flooding, or port mirroring on switches.

6.2.1 How Active Sniffing Works

In active sniffing, the sniffer sends packets or manipulates network behavior to capture traffic. For example, ARP spoofing involves sending fake ARP messages to associate the attacker's MAC address with the IP address of a legitimate device, intercepting its traffic.

6.2.2 Advantages of Active Sniffing

Capture on Switched Networks: Active sniffing can capture packets on switched networks, where passive sniffing is ineffective.

Comprehensive Analysis: It provides a more complete view of network traffic and can capture packets from all devices on the network segment.

6.2.3 Disadvantages of Active Sniffing

Intrusive: Active sniffing can affect network performance and alter data, potentially causing disruptions.

Detectable: It is easier for attackers to detect active sniffing, making it less suitable for stealthy monitoring.

7. Introduction to Wireshark

Wireshark is a free and open-source network protocol analyzer, regarded as the most widely used tool for network traffic analysis. Its popularity stems from its powerful features, user-friendly interface, and extensive protocol support. Developed initially as Ethereal by Gerald Combs in 1998, it was renamed Wireshark in 2006 due to trademark issues. Over the years, Wireshark has evolved into a comprehensive tool with contributions from a global community of developers.

7.1 History and Development

Early Days Gerald Combs developed Wireshark (originally named Ethereal) while working at a small ISP (Internet Service Provider). Combs needed a tool to troubleshoot network problems, but existing tools were either too expensive or lacked the necessary features. This motivated him to create his own network analyzer. Ethereal's initial release in 1998 quickly gained traction among network administrators and security professionals due to its robustness and flexibility.

Renaming and Growth

In 2006, the project was renamed Wireshark due to trademark issues. The renaming did not hinder its growth; instead, it marked a new era of development and expansion. Wireshark's open-source nature allowed for continuous enhancements and updates, ensuring it stayed at the forefront of network analysis tools. The global community of contributors has expanded its capabilities, added support for new protocols, and improved its usability.

Supported Platforms

7.2 Wireshark is available for multiple operating systems, ensuring broad accessibility:

Windows: Wireshark provides a Windows installer that includes the necessary libraries and components for seamless installation.

macOS: macOS users can download a dedicated installer package, which integrates Wireshark with the native operating system.

Linux: Wireshark is available through various package managers (e.g., APT for Debian-based distributions, YUM for Red Hat-based distributions) or can be compiled from source for customized installations.

7.3 Features of Wireshark

Wireshark offers a plethora of features, making it a powerful and versatile tool for network analysis:

7.2.1 Packet Capture

Wireshark captures live network traffic from multiple network interfaces, including Ethernet, Wi-Fi, and Bluetooth. It supports capturing data from both local and remote networks, providing flexibility for various use cases.

7.2.2 Detailed Packet Analysis

Wireshark provides a detailed breakdown of each packet, including headers, payload, and protocol information. Users can inspect every aspect of a packet, from its source and destination addresses to the data it carries. This granularity allows for in-depth analysis and troubleshooting.

7.2.3 Filtering Capabilities

Wireshark's filtering capabilities are among its most powerful features:

Display Filters: Allow users to view only the packets of interest based on criteria such as IP address, protocol type, or port number. Display filters are highly flexible and support complex expressions.

Capture Filters: Limit the packets captured to those that meet specific conditions, reducing the volume of data for analysis. Capture filters use a syntax similar to Tcpdump, providing a familiar interface for experienced users.

Color Coding

Packets in Wireshark are color-coded based on protocol type, making it easier to identify different types of traffic at a glance. Users can customize color rules to highlight specific packets or patterns, enhancing the visual representation of captured data.

Protocol Decoding

Wireshark supports an extensive number of protocols and can decode and display their data. It provides detailed information about each protocol layer, helping users understand the structure and content of network traffic. Wireshark's protocol decoders are continually updated to support new and emerging protocols.

Statistical Tools

Wireshark offers various statistical tools to analyze traffic patterns, protocol usage, and network performance:

Protocol Hierarchy: Displays the distribution of protocols in the captured traffic, providing insights into protocol usage.

Endpoint Statistics: Shows statistics for network endpoints, such as IP addresses and ports, helping identify active devices and their communication patterns.

IO Graphs: Visualize network traffic over time, aiding in the identification of trends and anomalies.

Export Options

Captured data can be exported in multiple formats for further analysis or reporting. Wireshark supports export to text, CSV, XML, and other formats, facilitating data sharing and integration

with other tools. This flexibility ensures that captured data can be used across different platforms and applications.

Using Wireshark

Using Wireshark involves several steps, from installation to capturing and analyzing network traffic:

Installation

Wireshark can be downloaded and installed from the official website (www.wireshark.org). The installation process is straightforward, with platform-specific instructions provided for Windows, macOS, and Linux. The installer packages include all necessary components, such as libraries and dependencies, ensuring a smooth setup experience.

Capturing Traffic

To capture network traffic, users select the network interface they want to monitor and start the capture process. Wireshark displays the captured packets in real-time, providing immediate insights into network activity. Users can stop the capture at any time and save the data for later analysis.

Applying Filters

Filters are essential for focusing on specific traffic of interest. Wireshark offers two types of filters:

Display Filters: Refine the view of captured packets based on criteria such as IP address, protocol type, or port number. Display filters are highly flexible and support complex expressions, enabling precise filtering of traffic.

Capture Filters: Limit the packets captured to those that meet certain conditions, reducing the volume of data for analysis. Capture filters use a syntax similar to Tcpdump, providing a familiar interface for experienced users.

Analyzing Packets

Captured packets are displayed in a list, and users can click on individual packets to view detailed information. The packet details pane shows the structure of each packet, including headers and payload. The packet bytes pane displays the raw data in hexadecimal and ASCII formats, allowing for in-depth analysis.

Saving Captures

Capture files can be saved in Wireshark's native format (.pcapng) or exported to other formats for further analysis. Saved captures can be opened later, shared with colleagues, or imported into other tools for additional processing.

Practical Applications of Wireshark

Wireshark has numerous practical applications in various fields, from network troubleshooting to security analysis and protocol development:

Network Troubleshooting

Network administrators use Wireshark to diagnose and resolve network issues. By analyzing captured packets, they can identify problems such as network congestion, protocol errors, and configuration issues. Wireshark helps pinpoint the root cause of network problems, facilitating quick and effective resolution.

Security Analysis

Wireshark is a valuable tool for security analysts in identifying potential threats and vulnerabilities. It can detect unusual traffic patterns, unauthorized access attempts, and malware activity. Security professionals use Wireshark to investigate security incidents, analyze attack vectors, and develop mitigation strategies.

Protocol Development

Developers use Wireshark to test and debug network protocols during development. By capturing and analyzing traffic generated by their applications, they can verify protocol compliance, identify bugs, and optimize performance. Wireshark's extensive protocol support and detailed analysis capabilities make it an essential tool for protocol development.

Educational Purposes

Wireshark is widely used in academic settings to teach students about network protocols and traffic analysis. It provides a hands-on learning experience, allowing students to explore real network traffic and understand the intricacies of protocol behavior. Educators use Wireshark to demonstrate networking concepts and practical troubleshooting techniques.

Advanced Features of Wireshark

Wireshark also offers several advanced features for more in-depth analysis and specialized use cases:

Protocol Dissector Plugins

Wireshark supports custom protocol dissector plugins, allowing users to extend its functionality and add support for proprietary or custom protocols. Developers can create plugins in C or Lua, integrating seamlessly with Wireshark's existing protocol decoders.

Remote Packet Capture

Wireshark can capture traffic from remote network interfaces using the Remote Packet Capture Protocol (RPCAP). This feature is useful for monitoring network segments that are not directly accessible from the user's machine. RPCAP allows Wireshark to capture traffic over the network, providing flexibility for distributed network environments.

Decryption Support

Wireshark can decrypt encrypted traffic, such as SSL/TLS, if the necessary keys are available. This capability is crucial for analyzing secure communication channels and identifying potential security issues. Wireshark supports several methods for decrypting SSL/TLS traffic, including key logging, RSA private keys, and pre-master secret keys.

Custom Profiles

Wireshark allows users to create custom profiles, which include personalized settings, filters, and color rules. Custom profiles enable users to tailor Wireshark to their specific needs and workflows, enhancing productivity and efficiency.

CHAPTER 2

Literature Survey

1. Bindu Dodiya. et al. [1] developed a method with the utilization of Wireshark, an efficient open-source packet analysis tool, for tracing and categorizing various attack signatures in network forensics. Wireshark's ability to capture live data at a microscopic level allowed administrators to identify malicious online behavior, detect data breaches, and reveal indicators of compromise for malware. The advantage lay in Wireshark's comprehensive analysis, providing a detailed understanding of network packets and enabling proactive measures to enhance cybersecurity. However, despite its effectiveness in uncovering a wide range of security threats, Wireshark had limitations as it lacked intrusion detection capabilities. Unlike dedicated systems, Wireshark did not offer real-time warnings or actively prevent unauthorized activities, underscoring the importance of implementing complementary security measures for proactive threat detection and response.
2. Giovanni Barbieri. et al. [2] implemented a comparative analysis method, contrasting Shodan-only assessments with large-scale traffic analysis at an Internet Exchange Point (IXP) via sFlow sampling. This approach, utilizing sFlow sampling, allowed the identification of Industrial Control Systems (ICS) endpoints engaged in genuine industrial traffic, overcoming Shodan's limitations. The methodology not only detected scanning activities but also differentiated between industrial and IT traffic, providing a more comprehensive understanding of insecure industrial protocol usage. Despite its advantage, the study's limitation was the reliance on a 31-day sampled traffic capture, potentially missing transient industrial traffic patterns. Furthermore, while effective in identifying legitimate industrial traffic, the proposed analytic framework might not address real-time threats or evolving cyber threats adequately, potentially limiting its immediate threat detection capabilities.
3. Muhammad Farrid Affiq Harirul Kamal. et al. [3] developed a dynamic Android botnet detection method using network analysis, offering real-time insights into application behavior. Extracting five features from Wireshark and SSL Packet Capture, the approach demonstrated promising accuracy, ROC value, and low FP value when tested with the Artificial Neural Network (ANN) algorithm. However, the study's limitation was its reliance on datasets from APKPure, Github, and Koodous, potentially limiting the representation of diverse Android applications and botnet behaviors. Despite achieving positive results, the proposed network traffic features may have had constraints in capturing nuanced variations in Android botnet characteristics, suggesting the need for further refinement and exploration in subsequent research.

4. Sujith Bebortta. et al. [4] implemented a method addressing the challenge of network traffic administration for diverse IoT devices. The focus was on efficiently characterizing inter-arrival rates through packet-level and flow-level analysis, facilitating the crucial identification and management of IoT devices for stable network activities and enhanced cybersecurity. The approach provided a precise understanding of network flows and insights into the strengths, weaknesses, and future scopes of state-of-the-art technologies for managing the expanding IoT landscape. However, a limitation emerged in the absence of specific details on proactive measures for identifying and isolating network vulnerabilities, warranting further exploration in subsequent research.
5. Ali Siddiqui. et al. [5] developed a method utilizing Wireshark as a network protocol analyzer for forensic analysis on network security attacks. Wireshark facilitated ethical hackers in collecting and analyzing data to uncover evidence of network intrusions, exposing vulnerabilities in cyber security at the user level. The advantage of Wireshark's functionality as a sniffing network tool allowed live capture and breakdown of network transmissions and various packets. This facilitated a detailed analysis of protocols like HTTP, TCP, and UDP, enhancing the understanding of network activity and identification of website vulnerabilities. However, a limitation in this work was the focus on classifying websites as secure or vulnerable solely based on Wireshark's packet sniffing, potentially oversimplifying the evaluation of website security.
6. Mohammed AL Fawar et al [6] presented the significance of traffic analysis for optimizing performance, identifying network anomalies and securing network, while tools such as Wireshark are referred for traffic analysis, network, penetration testing, etc., It’s about appreciating the significance of network analysis and the security problems that are associated with it.
7. Kovstur Maxim et al [7] presented a method for analyzing wireless network traffic which was written in python programming language and used panda library. In order to make wireless networks available and fault-tolerant, constant monitoring is needed, in that, the actual target is about making better tracking of anomalous traffic during analysis.

8. Laura Chappell et al [8] proposed a distributed network analysis in which traffic is inspected at multiple places on the network. The conventional way of doing this is to use a full-blown network Analyzer, which is quite expensive. Exported packet

record files are imported to real-time network analysis tool TOPAS and analyzed using the open-source network analyzer Wireshark.

9. Banerjee, Usha et al. [9] described Wireshark as a sniffing tool in the networks which is shown by an experimental setup where effectiveness of malicious packet detection in any network is shown. Inferences have been derived and the capabilities have been made evident which indicates that it could be a potential subject for future development into a robust intrusion detection system.

10. A Dabir et al [10] discusses the bottlenecks of commodity hardware-based packet capturing for local area networks (LANs) without data loss. Tests were run with a wireshark packet sniffer to capture packets directly from disk in Fast Ethernet networking with different setups. These were experiments that created large packets near line rate. Also play around with different kernel-level buffer sizes associated with the packet capturing socket.

11. Vixens Ndatinya Zhifeng Xiao et al [11] presented Network Forensics Analysis Using Wireshark. As the variety and volume of networked computing systems have been increasing, network security has become an increasingly crucial field due to the increasing number of attacks. Now the network administrators have to be able to monitor and analyze the network traffic to identify the event and the network administrator must be able to respond immediately to any event.

12. Piyush Goyal et al [12] Started a Comparison study of two of the best packets sniffing tools —Tcpdump and Wireshark. The increase in the sphere of the Internet has also broadened the spectrum of Networking, data transfer, and data security. They have become the preferred tools of the hackers to be exploited to scan for specific networks and eavesdrop on unencrypted information. These tools allow White Hat hackers to thwart the criminals by filtering out malicious packets and their source.

13. Samer Hamdani et al [12] reported research on a COAP vs. MQTT comparison study. IOT technology contains continuous data emitters or sensor data, predominantly transmitted by internet connection, generating homogeneous and massive data transmission. A low-cost method for distributed network analysis was proposed by Gerhard Munz et al [13]. In contrast, instead of having very expensive network analyzers at every observation point, the authors recommend using packet data exports from network devices supporting PSAMP and Flexible NetFlow. The idea is to collect packet dumps from these devices and export them to TOPAS, an open-source real-time network analysis framework, which then uses the network analyzer Wireshark to do in-depth analysis. With a Monitor Manager in place, only the necessary data for desired analysis purpose is exported. Consequently, it provides a practical approach that makes use of existing infrastructure and open-source tools to provide scalable and cost-effective distributed network analysis, while highlighting its merits and drawbacks.

14. Pallavi Asrodia et al [14] described the importance of network traffic analysis in the context of the increasing growth and complexity of computer networks. They laid emphasis on packet sniffers, so that they could efficiently manage, operate and monitor these networks and maintain their smoothness, ensuring their economic efficiency. Packet sniffing is a fundamental tool in network monitoring, troubleshooting, and traffic logging, for wired as well as wireless networks. It also goes into the basics about packet sniffers and how they work to analyze network traffic.

CHAPTER 3

Problem Statement

Objective:

To analyze and monitor network traffic in a given network environment to identify potential issues such as network congestion, security threats, and performance bottlenecks using the packet sniffer tool, Wireshark.

Background:

In today's digital age, network performance and security are paramount for businesses and individuals alike. With the increasing complexity of network infrastructures, it becomes crucial to have robust tools and methodologies for monitoring and analyzing network traffic. Wireshark, a powerful packet sniffer tool, provides detailed insights into network traffic, making it an essential tool for network administrators and security professionals.

Problem Description:

The goal of this project is to perform a comprehensive analysis of network traffic in a controlled environment using Wireshark. The analysis aims to identify key metrics such as bandwidth usage, traffic patterns, and potential security threats. By capturing and examining network packets, we can gain valuable insights into the network's performance and security posture.

Scope of Work:

Setup and Configuration:

- Install and configure Wireshark on the network.
- Define the network environment to be monitored (e.g., a small office network, a university campus network, etc.).

Data Collection:

- Capture network traffic over a specified period.
- Ensure diverse traffic by including activities such as web browsing, file transfers, video streaming, and online gaming.

Traffic Analysis:

- Analyze captured data to identify normal and abnormal traffic patterns.
- Measure bandwidth usage and identify the top bandwidth-consuming applications and devices.
- Detect potential security threats such as unauthorized access attempts, malware, and suspicious communication patterns.

Performance Evaluation:

- Assess the overall network performance and identify bottlenecks.
- Evaluate the impact of different types of traffic on network performance.

Reporting:

- Compile a detailed report summarizing findings, including visualizations such as graphs and charts.
- Provide recommendations for optimizing network performance and enhancing security based on the analysis.

Deliverables:

- A comprehensive report detailing the methodology, analysis, findings, and recommendations.
- Visualizations of network traffic patterns and key metrics.
- A step-by-step guide on using Wireshark for network traffic analysis.

Expected Outcomes:

- Enhanced understanding of network traffic patterns and performance.
- Identification of potential security threats and vulnerabilities.
- Actionable insights for optimizing network performance and security.

Tools and Technologies:

- Wireshark for packet capture and analysis.
- Additional tools for data visualization and reporting (e.g., Excel, Python scripts).

CHAPTER 4

Objectives

Study of Network Protocols:

To gain an in-depth understanding of network protocols such as TCP/IP, UDP, HTTP, FTP, and DNS.

To analyze the structure and functionality of network packets.

Evaluation of Packet Sniffer Tools:

To investigate and evaluate different packet sniffer tools, including Wireshark and tcpdump.

To learn the installation, configuration, and effective usage of these tools.

Network Traffic Capture:

To capture live network traffic using a packet sniffer tool.

To analyze the captured network traffic data for better comprehension of network communication.

Traffic Pattern Identification and Analysis:

To identify common traffic patterns in the network.

To analyze the traffic for anomalies or unusual patterns indicating network issues or security threats.

Network Performance Evaluation:

To assess network performance by analyzing packet loss, latency, and throughput.

To identify network bottlenecks and propose solutions for performance improvement.

Security Analysis and Implementation:

To identify potential security vulnerabilities through packet analysis.

To propose and implement security measures such as encryption, firewall configurations, and intrusion detection systems.

Custom Filter Development:

To develop custom filters for isolating specific types of traffic or protocols.

To conduct in-depth analysis on targeted network activities using these filters.

Report and Visualization Generation:

To develop methods for presenting analyzed data through comprehensive reports and visualizations.

To communicate findings effectively to stakeholders such as network administrators and security teams.

Compliance with Legal and Ethical Standards:

To understand and adhere to legal and ethical considerations related to network monitoring and data privacy.

To ensure compliance with relevant laws and organizational policies.

CHAPTER 5

METHODODLOGY

In this, proposed method for network traffic analysis using Wireshark encompasses a systematic and comprehensive approach to extract meaningful insights from network packets. Utilize Wireshark’s packet capturing capabilities to collect network traffic data at strategic points within the network, such as routers or switches. Ensure adequate packet sampling for accuracy.

Data Filtering:

Employ Wireshark's advanced filtering options to eliminate noise and isolate packets of interest. Filters may include IP addresses, port numbers, or specific protocols, depending on the analysis goals.

Packet Decoding:

Leverage Wireshark's deep packet inspection capabilities to decode packets and extract protocol-specific information. This step helps in understanding the nature of the traffic, such as HTTP requests or DNS queries.

Flow Analysis:

Group packets into flows or connections, considering source-destination pairs and their respective protocols. This aids in tracking communication patterns and identifying abnormal behavior.

Statistical Profiling:

Calculate various statistical metrics, including packet counts, packet size distributions, and traffic patterns over time. Wireshark provides tools for statistical analysis and graph generation.

Protocol Analysis:

Perform protocol-specific analysis, such as identifying protocol-specific anomalies or detecting irregularities in protocol handshake processes (e.g., TCP SYN-ACK).

Behavioral Analysis:

Employ heuristic or machine learning-based methods to identify abnormal network behavior. Train models on historical data to detect deviations from expected traffic patterns.

Security Signature Matching:

Utilize Wireshark's signature-based detection features to identify known attack patterns or malware signatures within the captured traffic. Implement an alerting system that triggers notifications for detected anomalies or potential security threats. Wireshark can be integrated with alerting mechanisms to automate this process. Create visual representations of network traffic patterns, including time-series graphs, heatmaps, and network topology diagrams, using Wireshark's visualization capabilities.

Forensic Analysis:

For security incidents, utilize Wireshark's ability to save packet capture files for later forensic analysis. This step is crucial for post-incident investigations.

Reporting:

Generate detailed reports summarizing the findings, including observed anomalies, security threats, and recommendations for network optimization and security enhancement. Implement continuous network traffic monitoring using Wireshark or compatible tools to ensure ongoing network security and performance.

This comprehensive method harnesses Wireshark's rich feature set and analytical capabilities to provide network administrators and security professionals with a powerful toolset for enhancing network visibility, security, and performance. It enables proactive threat detection, rapid incident response, and data-driven decision-making in the dynamic landscape of network management and cybersecurity.

CHAPTER 6

Algorithm

1. Setup

- Install Wireshark on the analysis machine.
- Connect the machine to the network you want to analyze.

2. Capture Network Traffic

- Open Wireshark.

- Select the appropriate network interface to capture packets.
- Start the packet capture.
- Specify a capture filter (if necessary) to capture only relevant traffic. For example, to capture only HTTP traffic, use the filter tcp port 80.

3. Filtering Captured Traffic

- Once the traffic is captured, apply display filters to focus on specific types of traffic.

Example filters:

- http for HTTP traffic.
- ip.addr == 192.168.1.1 to filter traffic to/from a specific IP address.
- tcp.port == 80 to filter traffic on a specific port.

4. Analyzing Traffic

Identify Communication Patterns:

- Analyze source and destination IP addresses to identify communication patterns.
- Identify frequent communicators and high-traffic nodes.

Protocol Analysis:

- Analyze the distribution of protocols (e.g., HTTP, HTTPS, FTP, DNS).
- Identify any unusual or unauthorized protocols.

Performance Metrics:

- Calculate metrics such as packet count, byte count, and throughput for different traffic types.

Latency and Response Time:

- Measure the round-trip time (RTT) and response time for TCP connections.

Error Analysis:

- Identify packet loss, retransmissions, and errors.

Security Analysis:

- Detect potential security threats like port scanning, unusual traffic patterns, or suspicious connections.
- Identify clear-text transmission of sensitive data.

5. Report Generation

Summarize the analysis results.

Generate visualizations (graphs, charts) for better understanding.

Document findings and recommendations.

Detailed Steps and Implementation

Step 1: Setup

- Ensure that Wireshark is correctly installed and that you have the necessary permissions to capture traffic on the network interface.

Step 2: Capture Network Traffic

- Open Wireshark and select the network interface.
- Click on the "Capture Options" (gear icon) to set capture parameters if needed.
- Start the capture by clicking the "Start" button.

Step 3: Filtering Captured Traffic

- Use Wireshark's display filter bar to enter specific filters.
- For example, to filter HTTP traffic, type http and press enter.
- You can combine filters using logical operators (e.g., ip.addr == 192.168.1.1 and tcp.port == 80).

Step 4: Analyzing Traffic

Use Wireshark's built-in analysis tools:

- Statistics menu: Provides various statistics like protocol hierarchy, endpoint list, conversations, etc.
- Follow TCP Stream: Useful for examining complete conversations.
- IO Graphs: Visualize traffic patterns over time.

Scripted Analysis:

- Export captured data to CSV or other formats for further analysis using external tools like Python scripts.

Step 5: Report Generation

- Compile all the analysis results into a comprehensive report.

- Use tools like Excel or matplotlib (Python) to create visualizations.

CHAPTER 7

Implementation

Data Filtering:

Employ Wireshark's advanced filtering options to eliminate noise and isolate packets of interest. Filters may include IP addresses, port numbers, or specific protocols, depending on the analysis goals.

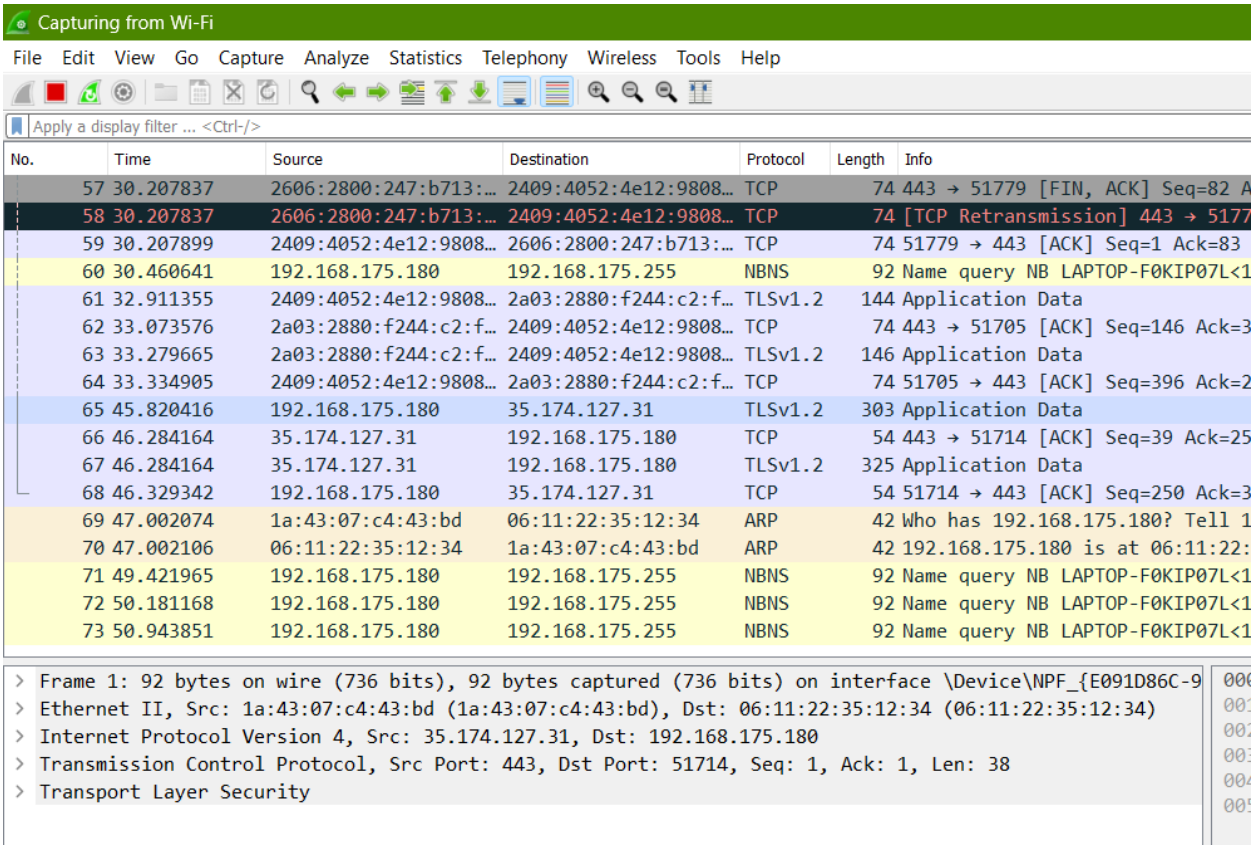

Fig. 7.1 Data filtering using Wireshark.

Packet Decoding:

Leverage Wireshark's deep packet inspection capabilities to decode packets and extract protocol-specific information. This step helps in understanding the nature of the traffic, such as HTTP requests or DNS queries.

Flow Analysis:

Group packets into flows or connections, considering source-destination pairs and their respective protocols. This aids in tracking communication patterns and identifying abnormal behavior.

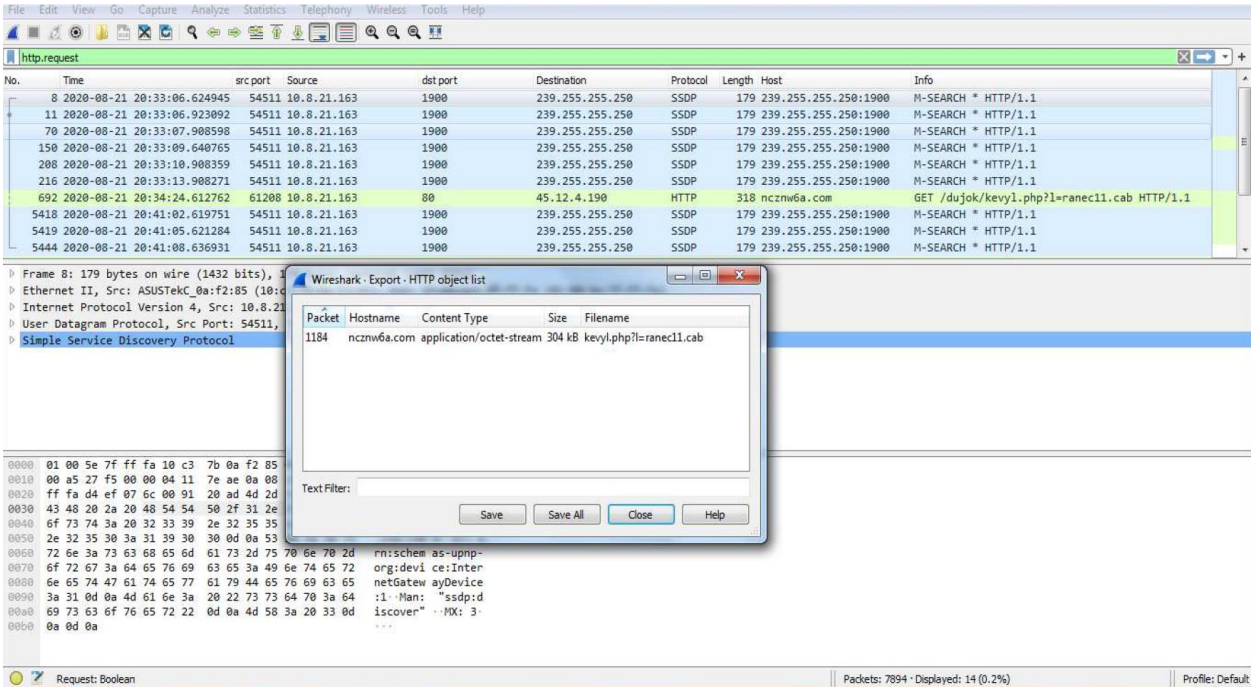

Fig. 7.2 Flow analysis using Wire Shark.

Statistical Profiling:

Calculate various statistical metrics, including packet counts, packet size distributions, and traffic patterns over time. Wireshark provides tools for statistical analysis and graph generation.

Protocol Analysis:

Perform protocol-specific analysis, such as identifying protocol-specific anomalies or detecting irregularities in protocol handshake processes (e.g., TCP SYN-ACK).

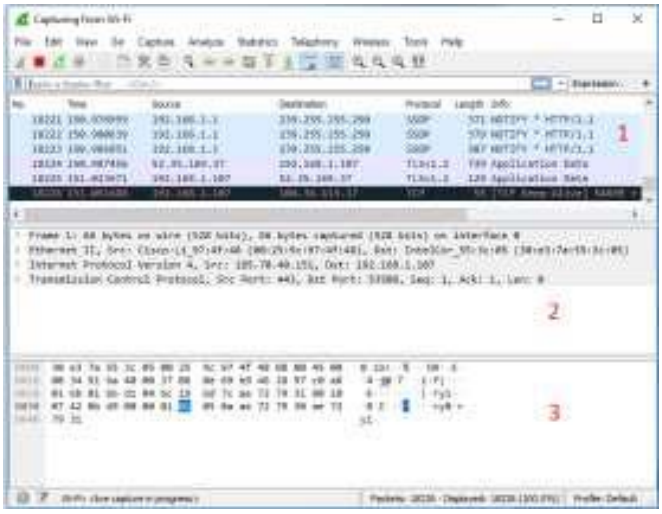

Fig. 7.3 Protocol analysis.

Security Signature Matching:

Utilize Wireshark's signature-based detection features to identify known attack patterns or malware signatures within the captured traffic. Implement an alerting system that triggers notifications for detected anomalies or potential security threats. Wireshark can be integrated with alerting mechanisms to automate this process. Create visual representations of network traffic patterns, including time-series graphs, heatmaps, and network topology diagrams, using Wireshark's visualization capabilities.

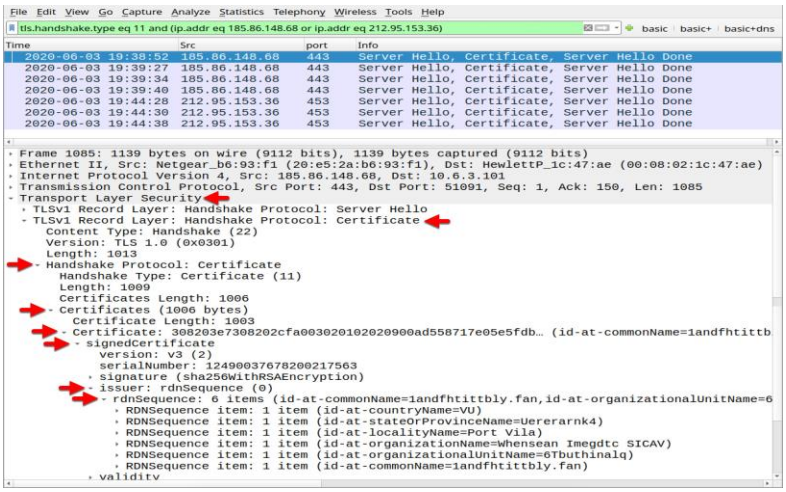

Fig. 7.4 Security Signature Matching.

Forensic Analysis:

For security incidents, utilize Wireshark's ability to save packet capture files for later forensic analysis. This step is crucial for post-incident investigations.

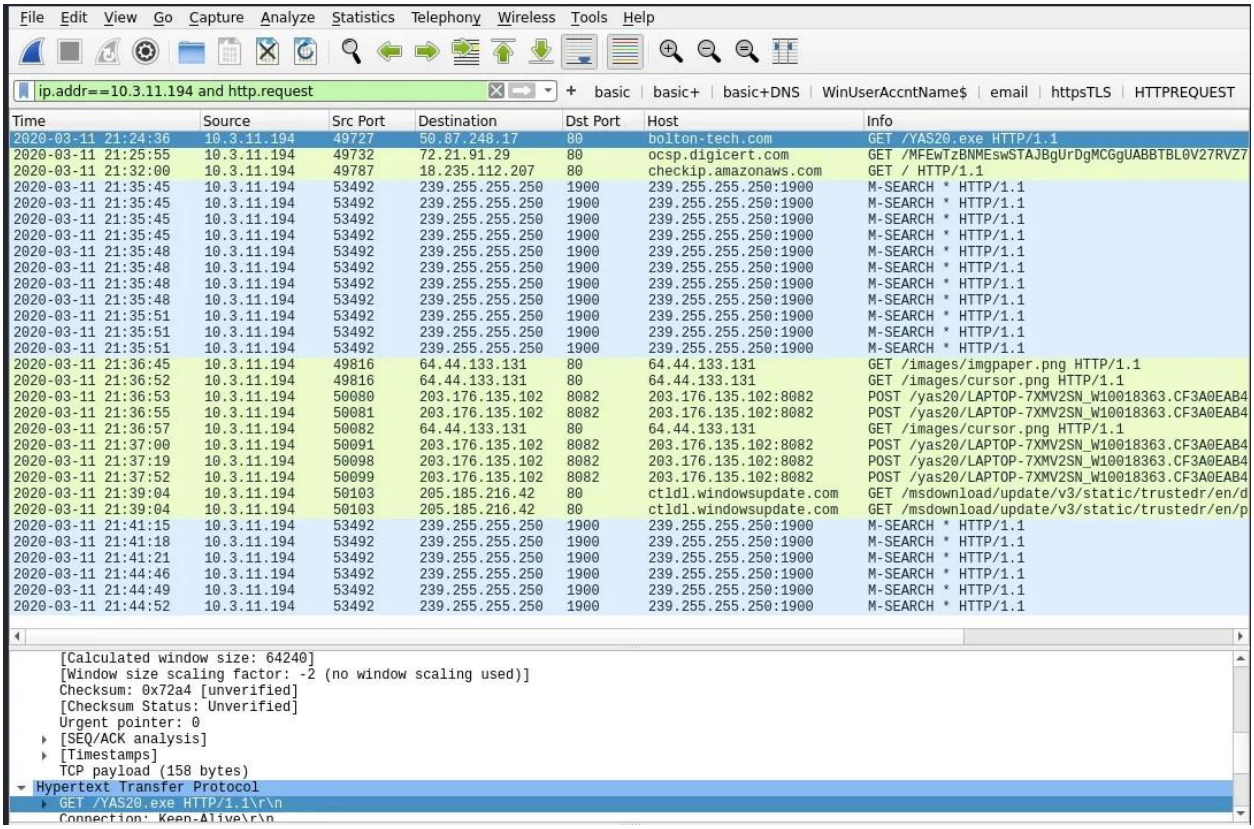

Fig. 7.5 Forensic analysis tab

CHAPTER 8

Result & Analysis

Network Traffic Overview

In this traffic, protocol filter of HTTP was applied, which resulted in showing the packets related to http protocol as shown in below figure.

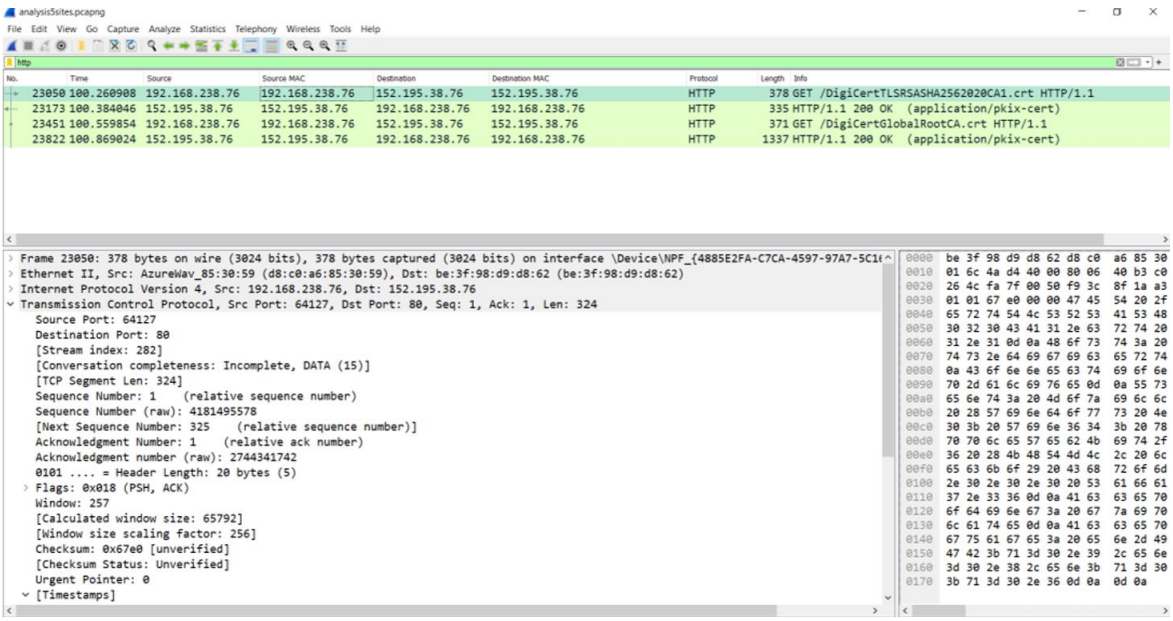

Fig. 8.1 Packet capturing

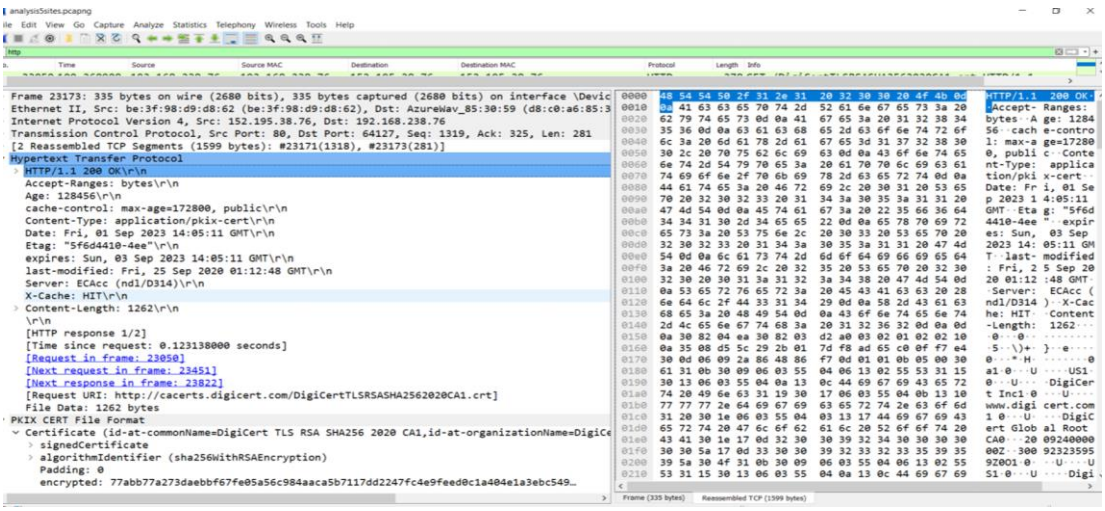

Fig. 8.2 http packet protocol details

Flow Graph: Flow graph shows the connection between the hosts. For each connection that was captured it shows the packet timing, direction, ports, and comments. It provides filters like ICMP (Internet Control Message Protocol) flows, ICMPv6 flows, UIM flows, and TCP flow . The flow graph window provides different controls based on that. With the help of flow graph you can easily figure out various port numbers and IP addresses and thus can easily get to get know if any unusual port number or IP address occurs in the traffic. The below figure shows the flow graph for the captured traffic.

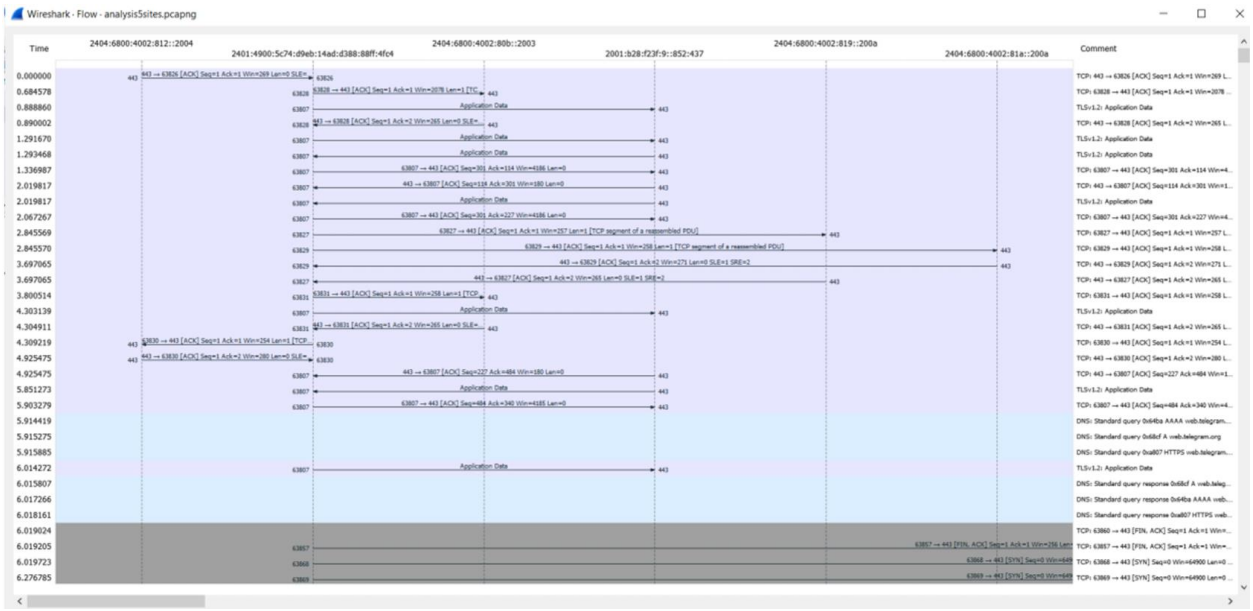

Fig. 8.3 Flow Diagram

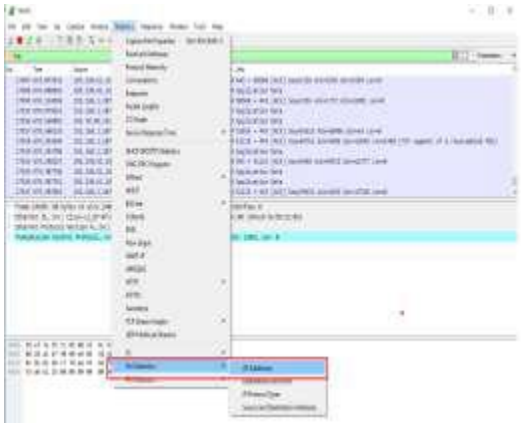

Fig. 8.4 Flow selection using wireshark.

Traffic Volume Analysis:

IO GRAPH: Display the number of packets or the amount of bytes per second for all packets that match the chosen filter.

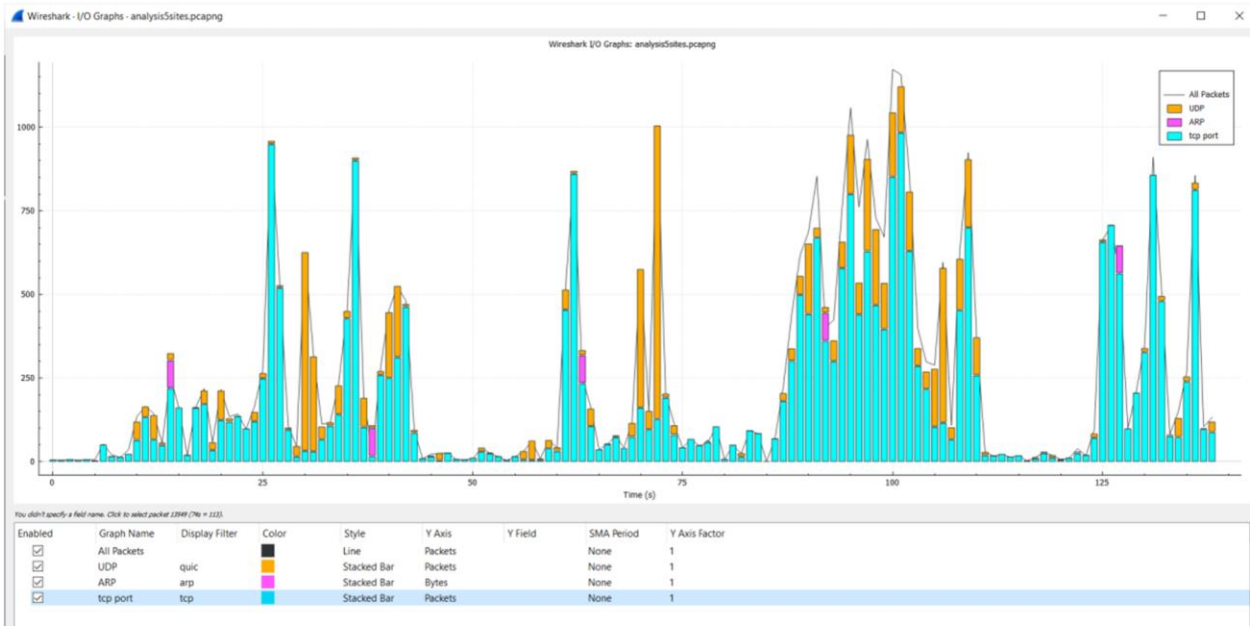

Fig 8.5 I/O Graph

HTTP -> Packet Counter: The packet counter the data regarding the HTTP packets. From here, we can analyze if there was any redirection or any kind of error. It helps in knowing if any attack like DDOS attack place, or were any packets redirected to any other unusual address. From the below figure, we analyzed that all the packets have 2xx Response, indicating that all packets were transmitted successfully and no packet was dropped.

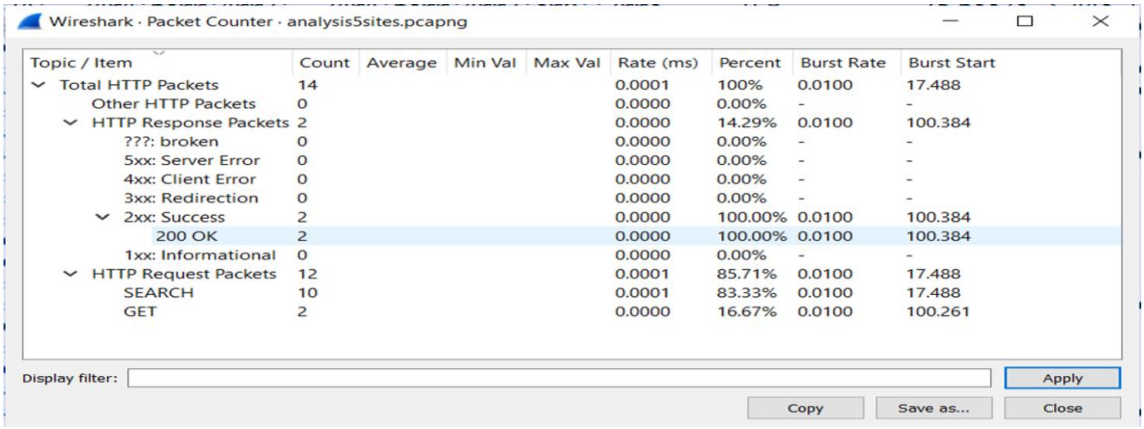

Fig. 8.6 Packet Counter

CHAPTER 9

Conclusion

Packet sniffing is useful to Analyze the data during the Transmission in the network. Sniffing tools are useful to implement it. It can be used for network traffic monitoring, traffic analysis, troubleshooting and other useful purposes. Packet sniffers can capture things like passwords and usernames or other sensitive information. Networks Sniffing in non-switched network is easy but sniffing in switched network is difficult because we use switches in network which narrow the traffic and send to particular system, so for sniffing in this type of network we use some methods. There are many available tools. Packet sniffer can be enhanced in future by incorporating features like making the packet sniffer program platform independent, and making tool by neural network Hence Sniffing should done in a manner to improve the performance of the network and to make it more secure.

Wireshark is useful for network monitoring, Traffic analysis and troubleshooting. By uploading Captured Packets (pcap file) on this tool you will get IP Addresses of given pcap and also the name of City, country and postal code related to them. With IP address management , organizations can track the status and availability of every device in their infrastructure. It is highly useful in preventing Cyber Threats(eg. Active attack) and assaults while ensuring a smooth workflow which enhances the Network Security.

CHAPTER 10

Future Scope

Enhanced Security Measures: With the increasing number of cyber threats, network traffic analysis using tools like Wireshark can help develop advanced security protocols. Future work can focus on integrating machine learning algorithms to detect anomalies and potential threats in real-time.

IoT Network Monitoring: As the Internet of Things (IoT) expands, there is a growing need for effective monitoring tools. Wireshark can be used to analyze the traffic of IoT devices, ensuring secure and efficient communication. Future projects could focus on optimizing Wireshark for IoT networks.

Performance Optimization: Network traffic analysis can help identify bottlenecks and optimize network performance. Future work can involve developing automated scripts or plugins for Wireshark that provide real-time performance insights and suggest improvements.

Big Data Integration: With the rise of big data, integrating Wireshark with big data analytics platforms could provide deeper insights into network traffic patterns. Future projects can explore the potential of such integrations to handle large-scale data and derive meaningful insights.

Cloud Network Analysis: As more organizations move to cloud-based infrastructure, analyzing network traffic in the cloud becomes crucial. Future research can focus on adapting Wireshark for cloud environments, ensuring it can handle the unique challenges posed by cloud networking.

Automated Incident Response: Future projects can aim to develop automated incident response systems using Wireshark. By integrating it with other security tools, it can help in the rapid detection and response to security incidents.

Compliance and Forensics: As regulatory compliance becomes more stringent, network traffic analysis can help in maintaining compliance and conducting forensic investigations. Future projects can focus on enhancing Wireshark's capabilities in these areas, making it easier to audit and investigate network activities.

REFERENCES

[1] Alfawareh Muhamed et al "A deeper Look into Network Traffic Analysis Using Wireshark" Academia.edu, 2015.

[2] K. Maxim et al “Research of wireless network traffic analysis using big data processing technology,” International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2021, pp. 115-121.

[3] Chappell et al “Wireshark network analysis: the official Wireshark certified network analyst study guide”, 2nd Edition, ISBN: 978-1- 893939-94-3, Chappell University, 2010.

[4] Usha Banerjee et al “Evaluation of the Capabilities of WireShark as a tool for Intrusion Detection” International Journal of computer applications, 2010, pp.1-5.

[5] A. Dabir et al “Bottleneck Analysis of Traffic Monitoring using Wireshark,” Innovations in Information Technologies (IIT), Dubai, United Arab Emirates, 2007, pp. 158-162.

[6] Ndatinya, Vivens et al. “Network forensics analysis using Wireshark” International Journal of Security and Networks, 2015, pp.91-106.

[7] P. Goyal et al “Comparative study of two most popular packet sniffing tools-Tcpdump and Wireshark,” International Conference on Computational Intelligence and Communication Networks (CICN), 2017, pp. 77-81.

[8] S. Hamdani et al “A Comparative study of COAP and MQTT communication protocols,” International Symposium on Digital Forensics and Security (ISDFS), 2019, pp. 1-5.

[9] G. Munz et al “Distributed Network Analysis Using TOPAS and Wireshark,” IEEE Network Operations and Management Symposium Workshops, 2008, pp. 161-164.

[10] Chappell, Laura et al “Wireshark 101: Essential skills for network analysis-wireshark solution series” Laura Chappell University, 2017.

[12] Asrodia, Pallavi et al “Network traffic analysis using packet sniffer” International journal of engineering research and applications, 2012, pp. 854-856.

[13] S. Sandhya et al “Assessment of website security by penetration testing using Wireshark,” International Conference on Advanced Computing and Communication Systems (ICACCS), 2017, pp. 1-4.

[14] R. Das et al “Packet tracing and analysis of network cameras with Wireshark” International Symposium on Digital Forensic and Security (ISDFS),2017, pp. 1-6.

[15] Aschin Dhakad et al “Real Time Network Traffic Analysis using Artificial Intelligence, Machine Learning and Deep Learning: A Review of Methods, Tools and

Applications" International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS),2023,pp.367-373.

[16] A. Vikram et al "Anomaly detection in Network Traffic Using Unsupervised Machine Learning Approach," International Conference on Communication and Electronics Systems (ICCES),2020, pp. 476- 479.

[17] A. Vikram et al "Blockchain Technology and its Impact on Future of Internet of Things (IoT) and Cyber Security," International Conference on Electronics, Communication and Aerospace Technology, 2022, pp. 444-447.

[18] V. Dharani et al "Spam SMS (or) Email Detection and Classification using Machine Learning," International Conference on Smart Systems and Inventive Technology (ICSSIT),2023, pp. 1104-1108.

[19] Upendra Shetty D R and A. Patil et al "Malicious URL Detection and Classification Analysis using Machine Learning Models," International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT),2023, pp. 470-476.

[20] BoYu "Based on Network sniffer implement network monitoring Computer Application and System Modeling (ICCASM) 2010 International Conference on Volume: 7,2010,Page(s):V7-1 -v7-3

[21] A Dabir,A Matrawy,"Bottleneck Analysis of Traffic Monitoring Using Wireshark" 4th International conference on Innovations in Information Technology,2007,IEEE Innovations 07,18-20 Nov.2007.Page(s) : 158-162

[22] Ashwani Kumar,Security Attacks in Manet - A Review,2011.

[23] F. Khan, R. Kothari, M. Patel and N. Banoth, "Enhancing Non-Fungible Tokens for the Evolution of Blockchain Technology," 2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), Erode, India, 2022, pp. 1148-1153, doi: https://doi.org/10.1109/ICSCDS53736.2022.9760849

[24] D.Madhavi, TCP Session Hijacking Implementation by Stealing Cook-ies,Vol. 2, Issue 11, 2015

[25] Ankita Gupta, Kavita, Kirandeep Kaur, Vulnerability Assessment and Penetration Testing, International Journal of Engineering Trends and Technology- Volume4Issue3-2013.

[26] D. Kothari, M. Patel and A. K. Sharma, "Implementation of Grey Scale Normalization in Machine Learning & Artificial Intelligence for Bioinformatics using Convolutional Neural Networks," 2021 6th International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2021, pp. 1071-1074, https://doi.org/10.1109/ICICT50816.2021.9358549.