RESEARCH PROPOSAL
(COIS71040)

DEVELOPING A PACKET SNIFFING APPLICATION TO REDUCE PACKET SNIFFING REDUNDANCY USING C PROGRAMING LANGUAGE

MSc in Computer Science

By

Muyideen Kazeem Oluwadare (21027842L)
Computer Science (Cyber Security)

Supervised By: Dr. Saeed

Staffordshire University

May 2022

Table of Contents

1. Introduction

Packet sniffing is the act of inspecting each packet as it moves over the network; it is a means by which an individual compromises information that belongs to other people on the network. A packet sniffer is a software application that is used to analyse packets that move over the network layer of the Transmission Control Protocol (TCP/IP) layer. The packet sniffer is a piece of equipment or software that network administrators legally employ to capture data frames exchanged across connected devices, and it is regarded as a vital monitoring tool for computer networks which some packet sniffing programs allow you to preserve data as auditing reports for later reference and evaluation (Ibrahim Diyeb et al., 2018). According to AL-Mukhtar Yasir Ahmed Abdullah (2008), sniffers operate because Ethernet was designed with the idea of sharing in mind, and that enable almost every network to use broadcast technology, which allows packets from one system to be read by another device in the network. All the other workstations will disregard the communication save by the one for whom it is intended, however, computers, on the other hand, may be programmed to take communications that are not intended for them. As explained by (Yang et al. (2018), it is critical to capture network traffic which may assist students in understanding how network transmission works and how data is packed into packets, transmitted, and directed to target destinations, moreover, network traffic behaviour was extensively studied in numerous research initiatives, like stepping-stone intrusion detection. Numerous research has been conducted on packet sniffing applications in different research areas such as comparing different packet sniffing applications, packet sniffing techniques, design a simplified packet sniffing application but none of them addresses the redundancy in packet sniffing applications based on the research carried out. This research work will focus on mitigating redundancy and removing the duplicate packets sniffed with other packets sniffing application by developing an enhanced and simplified user-friendly GUI packet sniffing application using the C programming language.

2. Research Background

Even though there are dozens of packet sniffers available in the marketplace to meet various industrial and personal needs, Tcpdump and Wireshark are the most extensively used and appreciated tools (Goyal & Goyal, 2017). Agreeing with Alina & Saraswat (2021), packet sniffer can be used alongside ARP spoofer to allow the attacker examine packets passing between faked endpoints A and B, which can be a command-line line (CLI) based raw script or a graphical user interface (GUI) package such as Wireshark and Tcpdump. Modern packet sniffers are highly broad and react to every packet equally; nevertheless, it may be specifically

dependent on the protocol type, e.g. TCP Sniffer, ARP Sniffer, and so on. As explained by Xu et al. (2020), sniffers are an indispensable tool, and repetitive sniffer implementation is popular; however, under ideal monitoring situations, it is probable to occur in multiple copies of the collected data packet, however, due to unreliable supervision scenarios, every sniffer can correctly determine data packets with a probability, such as a capture possibility which is described as the chances of effectively capturing data packets. It should be emphasized that the capturing rate indicates that the sniffers' abilities in network detection and may be studied before deployment in the system through a learning framework. Numerous technologies exists to track network traffic, typically, such programs will place a computer's network card into the full-duplex mode, allowing the computer system to listen to all activity in that area of the network and these packets can be filtered depending on the IP-related header data included in the packets which normally can filter this provided basic criteria for the IP addresses and ports available in the packets (Oluwabukola et al., 2013). According to D. Álvarez et al. (2021), network management based on packet sniffing is among the most successful methods used by system administrators and security professionals to uncover risks inside an internal network, but while appearing to be a simple operation, it may be a time-consuming and resource-intensive procedure. The packet sniffer application monitors the encoded data existing in the wired or wireless network. When the algorithm discovers relevant data, such as an email address or a username in a short sentence, it quickly apprehensions the data. Other identifications, such as cookie and session id, are feared by the software (Saroha, 2020). As stated by Poudél (2019), the commitment to monitoring tools has enhanced for security and network experts, who are heavily reliant on classical packet sniffer tools such as Wireshark and tcpdump. But nevertheless, the information supplied by such tools is really very huge, and even network professionals sometimes find it difficult to filter and acquire the necessary result. Furthermore, these industry-standard tools require solid knowledge of networking protocols, making them unsuitable for lay people and end-users. Following on a preliminary survey of related research, it indicates that perhaps the current packet sniffer application produces a repeating captured data packet, as well as a lack of structured result output, making data packet analysis more time consuming and exhausting. This research project will address the aforementioned setback by developing a packet sniffing application in C programming language to mitigate packet sniffing redundancy, and provide a formal highly specialised standard application programme interface (API), accompanied by a JavaScript Object Notation (JSON), a graphic user Interface (GUI) with a distinct captured data packet efficiency,

perform vulnerability analysis on the captured data packet, and mitigation techniques to prevent exploitation of the information.

3. Research Question

A research problem with extensive information was created to obtain appropriate and unbiased responses. The main question was, "What features should really be introduced, and what are the limitations of the present packet sniffing programme that should be modified to eliminate the redundancy in sniffing data packets for deep data inspection, and data packet analyses?"

4. Aim

This research study intends to examine relevant research on packet sniffing applications, analyse the current challenges, develop an enhanced packet sniffing software to produce a unique captured data packet efficiency that prevents redundant information in data packet capture, and evaluate the result produced against the existing specified requirements.

5. Objective

- To investigate similar work with packet sniffer programmes
- To investigate the issue of data packet redundancy.
- To create an API microservices that other packet sniffing applications can use.
- To unit test the proposed application and ensure compliance with standards.
- To assess the outcome using the existing packet sniffer application
- To generate output that can be fed into network mapping and vulnerability assessment software.

6. Methodology

The methodology to be used for the research project are as follows:

- **Research**: Numerous research would be carried out to determine the setback on the current packet sniffing application and how to mitigate or enhance their current features.
- Planning: The project duration and how to achieve the proposed project will be determine at this point.
- **Design:** The flow chart, algorithms, the functionality needed for the application would be determine, and the best design approaches will be adopted.

- **Development:** This is where the coding for the packet sniffing application will be done alongside with bug fixing.
- **Testing:** This is where the proposed packet sniffing software will be testing for errors to fix them and to ensure that it has really solve the problem discovered with other packet sniffing application
- **Setup:** The proposed develop packet sniffing application will be install on a machine connected to the network for the final testing and useability.
- **Maintenance:** A continuous maintenance will be carried out on the application to prevent it from being outdated and to improve it features for better and quality performance.

7. Deliverables

Following the completion of the research project, the following deliverables must be submitted:

- Gantt Chart/Project Plan displaying the entire duration of the research work
- Final Project Report
- Proposed developed application screenshots with user manual
- Technical documentation and source code
- A comparative analysis of current packet sniffing application with their current challenges
- A restful API endpoint that can be consume by other packet sniffing application, and software application/endpoints that needed a packet sniffing features in their application.

8. Academic Challenges

Academic difficulties include researching the most recent ways for reducing duplication in packet sniffing, as well as the existing packet sniffing application's inability to provide a software development kit that might be used to further investigate the packet intercepted. The biggest concern may be the time constraint, which is incredibly short due to the substantial research necessary. Another challenge would be to improve present programming language (C) expertise to construct the algorithm, therefore choosing a new suitable language for the algorithm, inventing the method, and implementing it may be challenging. Furthermore, the application's development process will be a time-consuming task.

9. Plan of work

The delivery schedule for the mid-point evaluation and the project work heavily influences the development strategy. To achieve the mid-point time frame, multiple work items must be initiated quickly to demonstrate progress and the proper direction. The key job for the final submittal will be constructing the artifact and conducting independent research around it. The study will be enhanced by critical examination and summarised in a summary, as well as a view on future research in this area. Aside from working on the project work and the artifact, two presentations should always be prepared: one for the mid-point assessment and one for the final oral. The below Gantt chart show the proposed research work duration which shows number of of weeks, and what is expected to be achieved for the completion of the project.

No of Weeks													
	1	2	3	4	5	6	7	8	9	10	11	12	13
Abstract													
Introduction, Aim, Objective, Hypothesis													
Literature Review													
Define Evaluation Criteria													
Design Requirement													
Validator for development specifications													
Developmental Case Study													
Mid-Point Review													
Testing													
Critical Evalutation													
Conclusion													
Final Work Presenation													

10. Resources

The following resources are required for the research project:

- Hardware Requirements
    - A personal computer with 16GB of RAM, Core i7 @ 2.5 GHz Quad-Core 64bit supported processor with hard drive of 512GB storage space.
    - A network router or switch
    - Internet Connection to download the require software framework, and to access research materials need for achieve the result of the research work
- Software Requirements

    - A Linux Virtual Machine (Kali or Parrot OS preferred) residing as a guest on the host machine started above
    - Windows / MAC operating system
    - Microsoft Visual Studio (2019 or 2022)
    - Wireshark, Nmap, Metasploit OpenVAS software
    - An API testing software application (Postman Preferred)
- Academic Materials
    - Related research conference papers and journals, and textbooks

11. References

Alina, A. & Saraswat, S. (2021). *Understanding Implementing and Combating Sniffing and ARP Spoofing; Understanding Implementing and Combating Sniffing and ARP Spoofing*.

AL-Mukhtar Yasir Ahmed Abdullah, M. (2008). *Developing a Sniffer Detector for Windows Operating Systems*.

D. Álvarez, P. Nuño, F. G. Bulnes & J. C. Granda (2021). *Performance_Analysis_of_Packet_Sniffing_Techniques_Applied_to_Network_Monitoring*.

Goyal, P. & Goyal, A. (2017). *Comparative Study of two Most Popular Packet Sniffing Tools-Tcpdump and Wireshark.*

Ibrahim Diyeb, I.A., Saif, A. & Al-Shaibany, N.A. (2018). Ethical Network Surveillance using Packet Sniffing Tools: A Comparative Study. *International Journal of Computer Network and Information Security*. [Online]. 10 (7). p.pp. 12–22. Available from: http://www.mecs-press.org/ijcnis/ijcnis-v10-n7/v10n7-2.html.

Oluwabukola, O., Oludele, A. & Ogbonna, A.C. (2013). *A Packet Sniffer (PSniffer) Application for Network Security in Java*.

Poudél, R. (2019). *Packet Sniffer to Sniff Sensitive Credentials Only Network Ports and Service Protocols View project Scripting Wireless Security Tools with Python & Scapy View project*. [Online]. Available from: https://www.researchgate.net/publication/339210096.

Saroha, S. (2020). *Restraining Packet Sniffing & Security: A Brief Overview*. [Online]. Available from: www.ijisrt.com955.

Xu, J., Gong, S., Zou, Y., Liu, W., Zeng, K., Niyato, D., Xu, J., Zou, Y. & Liu, W. (2020). Redundant Sniffer Deployment for Multi-Channel Wireless Network Forensics With Unreliable Conditions; Redundant Sniffer Deployment for Multi-Channel Wireless Network Forensics With Unreliable Conditions. *IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING*. [Online]. 6 (1). Available from: http://www.ieee.org/publications_standards/publications/rights/index.html.

Yang, J., Zhang, Y., King, R. & Tolbert, T. (2018). *Sniffing and Chaffing Network Traffic in Stepping-Stone Intrusion Detection; Sniffing and Chaffing Network Traffic in Stepping-Stone Intrusion Detection*. [Online]. Available from: www.tcpdump.org.