PCapper Technical Report

Ahmad Azeez

COMP3260_01 - Computer Network Security

Thompson Rivers University

Dr. Anthony Aighobahi

October 28th, 2024

Abstract

The purpose of this report is to present a tool, called **PCapper**, that uses Deep Packet Inspection (DPI) to identify potential network threats, including HTTP, SSH, and DNS tunneling. The tool uses packet capture and analysis techniques to enhance real-time security monitoring and offline security scanning.

The scope of the project includes:

- Implementing detection mechanisms for tunneling attacks.
- Adding real-time traffic analysis.
- Testing the system against real-world examples.

The findings demonstrate that the DPI approach is very effective in identifying suspicious patterns such as high-entropy DNS queries, unauthorized HTTP requests, and anomalous SSH traffic, without relying on machine learning models, which can be resource intensive. The tool successfully flagged malicious activities and generated the proper insights of the flagged packets so users can perform further inspections.

The report concludes that **PCapper** meets its objectives by providing a proof-of-concept for real-time threat detection, while highlighting challenges such as scalability and false positives handling. Key recommendations include extending the system's capabilities by making it so it also blocks or "quarantines" suspicious network traffic or domains, and reducing the amount of false positives the tool provides. This project successfully shows the importance of DPI in addressing modern cybersecurity challenges.

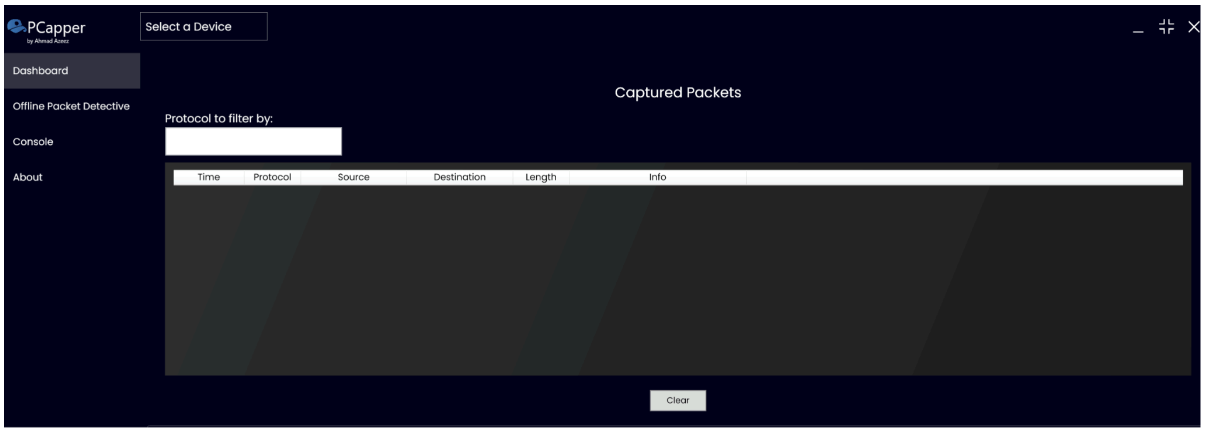


Table of Contents

Summary

The findings in this report demonstrate that DPI techniques, with additional tunneling detection implementations, successfully detect malicious patterns such as DNS, HTTPS and SSH tunneling, Trickbot malware, and encrypted traffic anomalies. By focusing on tunneling detection, the project demonstrates the effectiveness of DPI over the usual approaches, such as rule-based or machine learning methods, in detecting obfuscated malicious activities.

The report begins with an introduction that shows the purpose and significance of the project, followed by a literature review, particularly the work by Deri and Fusco (2021) on DPI in cybersecurity. The methodology describes the step-by-step implementation, including packet capturing, payload analysis, and tunneling detection algorithms. Results show the tool's ability to identify suspicious DNS, HTTPS, and SSH traffic patterns, with examples including high-entropy DNS queries and unauthorized HTTP/SSH requests. Challenges, such as handling encrypted traffic and minimizing false positives, are discussed.

The conclusion highlights the tool's success in achieving its objectives as a proof-of-concept tool while recommending future enhancements, such as supporting additional protocols, and implementation of a "blocking" system. The report shows the importance of DPI in modern cybersecurity.

Introduction

The PCapper project aims to provide a robust solution for monitoring and securing networks by detecting unauthorized requests and suspicious activities in real time and offline. With increasing sophistication in network threats, the usual or traditional tools often fail to detect encrypted malicious traffic. This project addresses these problems by implementing DPI techniques with additional DNS, HTTPS, and DNS tunneling detection techniques to analyze network packets deeply.

Background Information

This idea came up when an article titled “Using Deep Packet Inspection in CyberTraffic Analysis”*(Deri & Fusco, 2021)* was found and read. The idea of refusing to use resource intensive machine learning models, and the use of a DPI system instead, was fascinating. Another article titled “What Is DNS Tunneling and How to Detect and Prevent Attacks”*(Dizdar, 2023)* was discovered, which showed how dangerous and tricky tunneling attacks are. It turned out that the DPI algorithms used in the paper were not good enough at detecting advanced tunneling attack techniques, so the idea of the creation of a tool that implements the discussed DPI algorithms with advanced tunneling attack detection algorithms was born.

Specific Objectives:

- Develop a DPI-based system/tool that identifies DNS, HTTPS, and SSH tunneling within network traffic.
- Improve network security in the domain of encrypted traffic inspection.
- Complete the project within a 1-month timeframe.
- Test the system/tool with real world attacks/examples.
- Gain a thorough understanding of detection algorithm creation and enhancement. Especially within the context of DPI and tunneling detection methods.
- Create a user friendly and visually appealing application.

A logo designed for the app to make it more visually appealing.

Literature Review

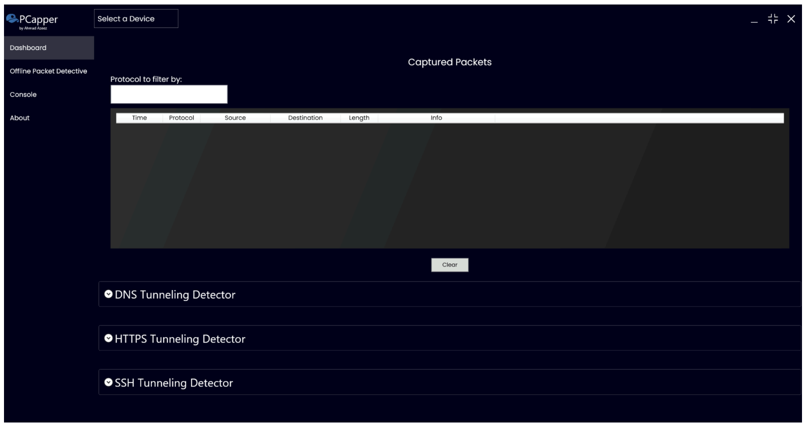
I will be using the paper “Using Deep Packet Inspection in CyberTraffic Analysis” by Luca Deri and Francesco Fusco (2021) as a foundation. This paper discusses the application of DPI for effective threat detection and highlights the challenges posed by encrypted traffic. It emphasizes the importance and need for better techniques to analyze traffic patterns without compromising privacy, making it a great paper to base my project off of.

The system in the paper extracts network traffic features like protocol types, periodic communication patterns, and suspicious domain names. This enables the system to spot weird or malware-infected traffic. I especially love the decision to use a DPI system instead of a machine learning algorithm, which is the usual approach for network analysis nowadays. This decision was made due to the lack of annotated data, computational intensity, and unexplainable predicted outcomes of a machine learning model.

Methodology

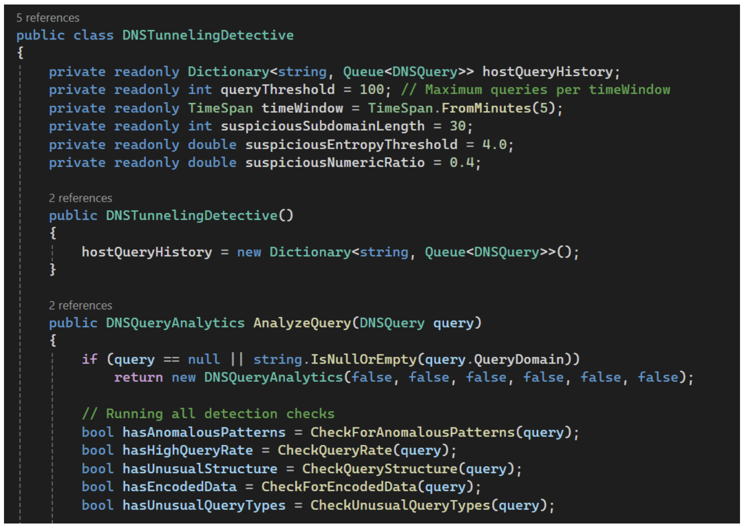
This is the approach that was used in completing this project:

- Reviewed and studied recent research on DPI and tunneling detection.
- Acquired traffic datasets containing DNS, HTTPS, and SSH tunneling examples for testing from “Malware Traffic Analysis” *(Malware-Traffic-Analysis.net, 2019)*.
- Implemented algorithms to analyze packet payloads for suspicious patterns, including DNS tunneling and Trickbot, with the help of an article titled “What Is DNS Tunneling and How to Detect and Prevent Attacks” *(Dizdar, 2023)*.
- Simulated attacks using tools like **dnscat2** and python scripts to check detection accuracy.
- Designed a user-friendly interface to display flagged packets and provide actionable insights.

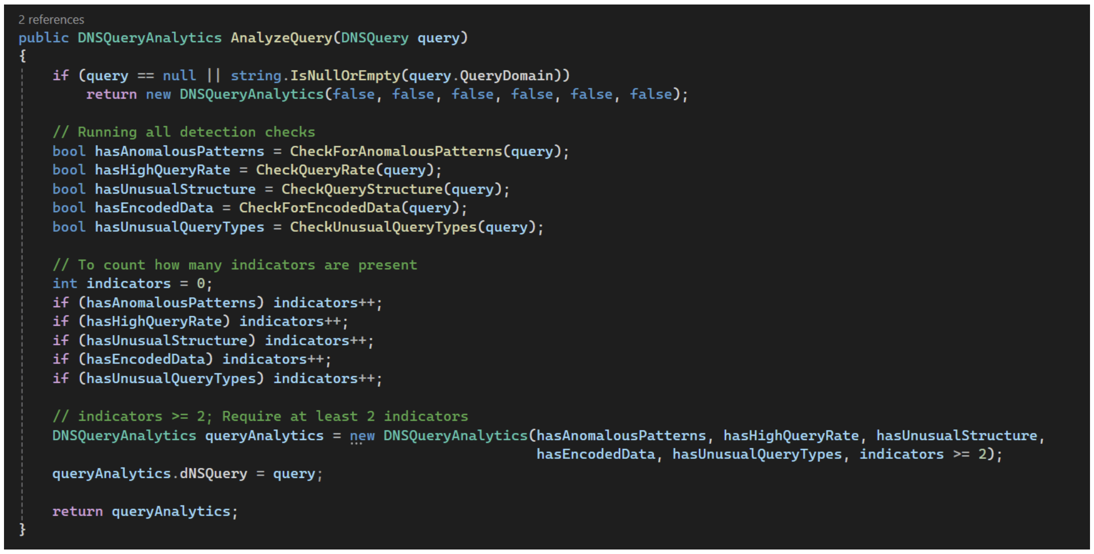


User interface for live packet analysis.

Example code for DNS Tunneling detection:



```
5 references
public class DNSTunnelingDetective
{
    private readonly Dictionary<string, Queue<DNSQuery>> hostQueryHistory;
    private readonly int queryThreshold = 100; // Maximum queries per timeWindow
    private readonly TimeSpan timeWindow = TimeSpan.FromMinutes(5);
    private readonly int suspiciousSubdomainLength = 30;
    private readonly double suspiciousEntropyThreshold = 4.0;
    private readonly double suspiciousNumericRatio = 0.4;

    2 references
    public DNSTunnelingDetective()
    {
        hostQueryHistory = new Dictionary<string, Queue<DNSQuery>>();
    }

    2 references
    public DNSQueryAnalytics AnalyzeQuery(DNSQuery query)
    {
        if (query == null || string.IsNullOrEmpty(query.QueryDomain))
            return new DNSQueryAnalytics(false, false, false, false, false, false);

        // Running all detection checks
        bool hasAnomalousPatterns = CheckForAnomalousPatterns(query);
        bool hasHighQueryRate = CheckQueryRate(query);
        bool hasUnusualStructure = CheckQueryStructure(query);
        bool hasEncodedData = CheckForEncodedData(query);
        bool hasUnusualQueryTypes = CheckUnusualQueryTypes(query);
```

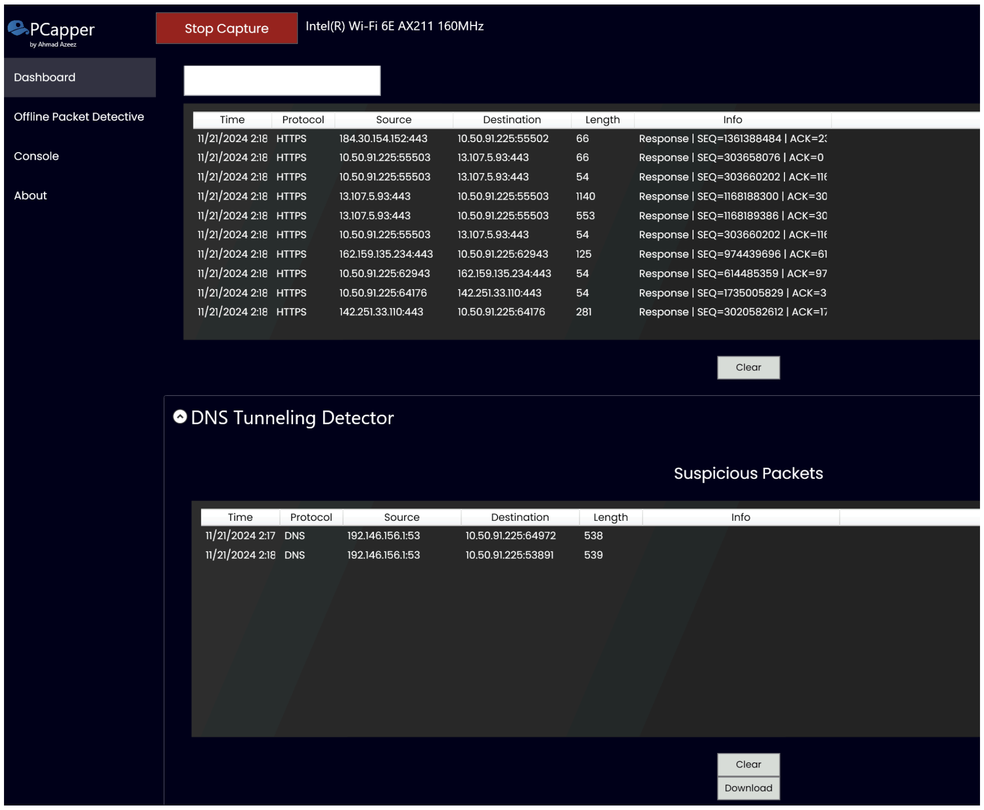


```
2 references
public DNSQueryAnalytics AnalyzeQuery(DNSQuery query)
{
    if (query == null || string.IsNullOrEmpty(query.QueryDomain))
        return new DNSQueryAnalytics(false, false, false, false, false, false);

    // Running all detection checks
    bool hasAnomalousPatterns = CheckForAnomalousPatterns(query);
    bool hasHighQueryRate = CheckQueryRate(query);
    bool hasUnusualStructure = CheckQueryStructure(query);
    bool hasEncodedData = CheckForEncodedData(query);
    bool hasUnusualQueryTypes = CheckUnusualQueryTypes(query);

    // To count how many indicators are present
    int indicators = 0;
    if (hasAnomalousPatterns) indicators++;
    if (hasHighQueryRate) indicators++;
    if (hasUnusualStructure) indicators++;
    if (hasEncodedData) indicators++;
    if (hasUnusualQueryTypes) indicators++;

    // indicators >= 2; Require at least 2 indicators
    DNSQueryAnalytics queryAnalytics = new DNSQueryAnalytics(hasAnomalousPatterns, hasHighQueryRate, hasUnusualStructure,
                                                             hasEncodedData, hasUnusualQueryTypes, indicators >= 2);
    queryAnalytics.dNSQuery = query;

    return queryAnalytics;
}
```

Results and Discussion

The tool successfully flagged unauthorized and suspicious packets. It identified suspicious domains and tunneling behaviors with high accuracy.

- **DNS Tunneling Detection:**
    - Identified high-entropy DNS queries and suspicious domain patterns, such as long subdomains and encoded data.
    - False positives were observed in cases of safe “high-entropy” queries, which were mitigated by refining detection thresholds.
- **HTTPS Tunneling Detection:**
    - Detected unauthorized HTTP requests using payload inspection.
    - There were some challenges with encrypted HTTPS traffic due to the inability to perform detailed content inspection, which required the reliance of metadata such as header and flow patterns.
- **SSH Tunneling Detection:**
    - Successfully flagged SSH sessions with anomalous traffic patterns, such as high data transfer rates and long session durations.
    - Detection of unauthorized SSH connections based on predefined IP/port rules was effective in simulated attacks.



Time	Protocol	Source	Destination	Length	Info
11/21/2024 2:18	HTTPS	184.30.154.152:443	10.50.91.225:55502	66	Response | SEQ=1361388484 | ACK=2
11/21/2024 2:18	HTTPS	10.50.91.225:55503	13.107.5.93:443	66	Response | SEQ=303658076 | ACK=0
11/21/2024 2:18	HTTPS	10.50.91.225:55503	13.107.5.93:443	54	Response | SEQ=303660202 | ACK=11
11/21/2024 2:18	HTTPS	13.107.5.93:443	10.50.91.225:55503	1140	Response | SEQ=1168188300 | ACK=30
11/21/2024 2:18	HTTPS	13.107.5.93:443	10.50.91.225:55503	553	Response | SEQ=1168189386 | ACK=30
11/21/2024 2:18	HTTPS	10.50.91.225:55503	13.107.5.93:443	54	Response | SEQ=303660202 | ACK=11
11/21/2024 2:18	HTTPS	162.159.135.234:443	10.50.91.225:62943	125	Response | SEQ=974439696 | ACK=61
11/21/2024 2:18	HTTPS	10.50.91.225:62943	162.159.135.234:443	54	Response | SEQ=614485359 | ACK=97
11/21/2024 2:18	HTTPS	10.50.91.225:64176	142.251.33.110:443	54	Response | SEQ=1735005829 | ACK=3
11/21/2024 2:18	HTTPS	142.251.33.110:443	10.50.91.225:64176	281	Response | SEQ=3020582612 | ACK=17

Time	Protocol	Source	Destination	Length	Info
11/21/2024 2:17	DNS	192.146.156.1:53	10.50.91.225:64972	538	
11/21/2024 2:18	DNS	192.146.156.1:53	10.50.91.225:53891	539	

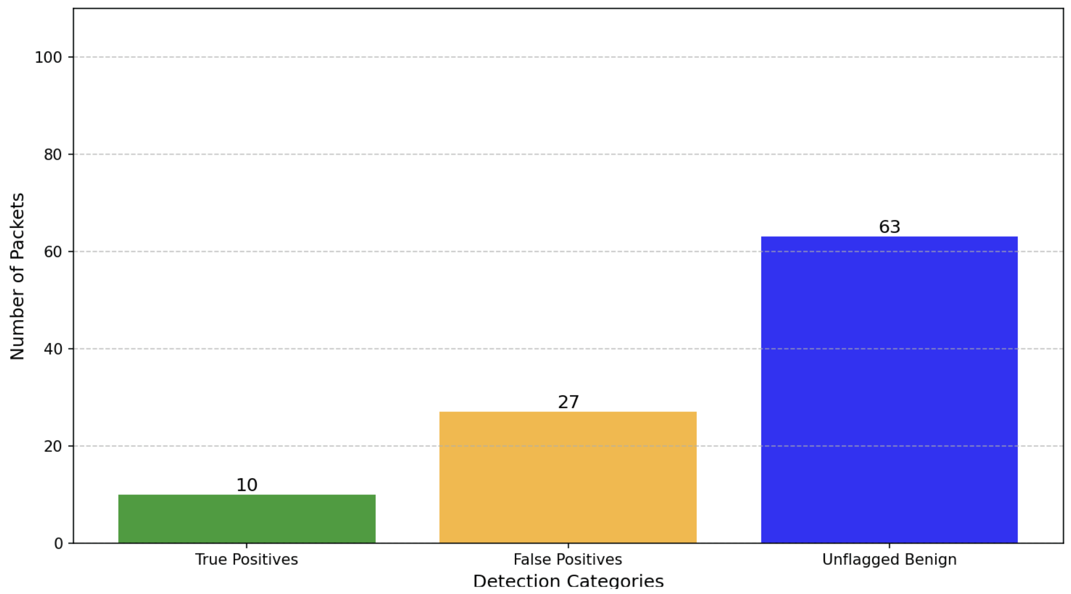
Analysis

The system shows strong capabilities in recognizing malicious traffic or packets while maintaining a balance between detection accuracy and computational efficiency. Unlike machine learning models, the DPI-based approach provided explainable and actionable insights, making it suitable for real-time traffic or packet analysis.

Unexpected Findings and Challenges

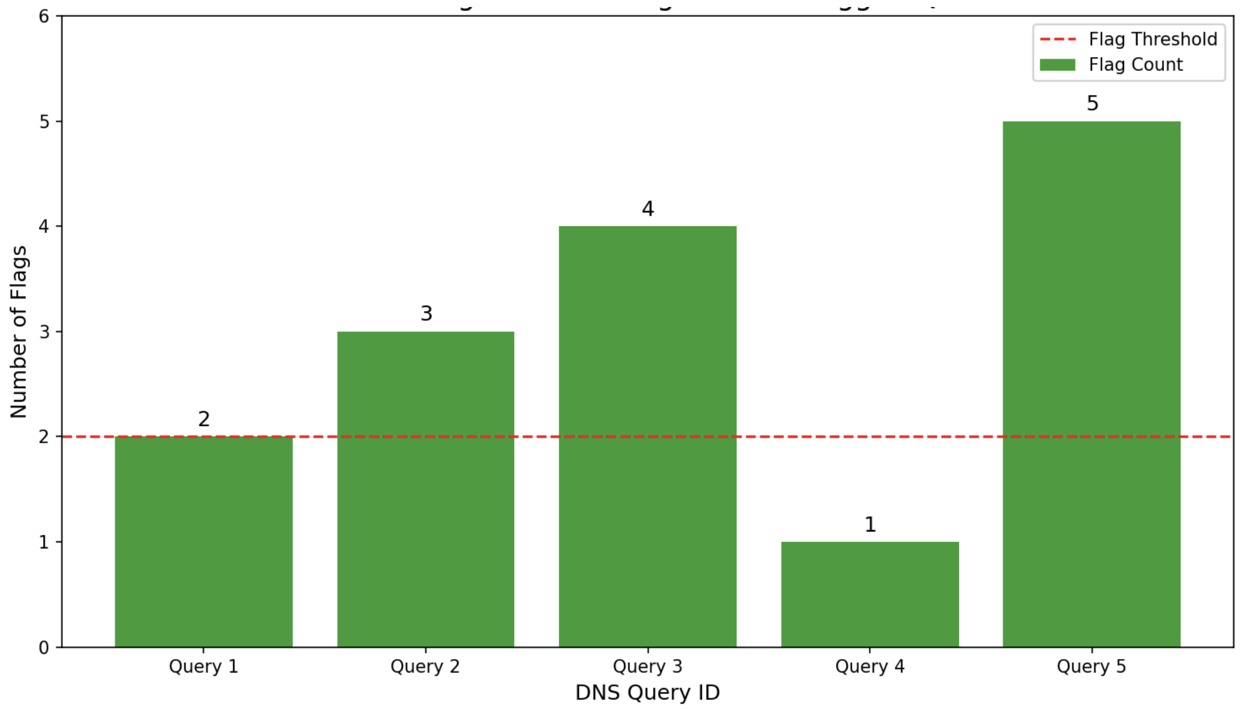
- **Encrypted HTTPS traffic:**
  - The tool was unable to inspect the payloads of HTTPS packets due to encryption, limiting detection to metadata and headers. This challenge is partly due to ethical concerns regarding content inspection.
- **False Positives:**
  - High-entropy DNS queries, though safe in some cases, were flagged as suspicious, requiring adjustments to detection criteria.
- **Scalability:**
  - The system performed well in small to medium network scenarios but latency was discovered when processing large datasets from large PCAP files, which highlights the need for optimization in packet parsing and processing.

DNS Tunneling Attack Detection Graph:



In the graph above, a total of 100 packets were scanned, 10 of which were malicious (packets with DNS tunneling attack activities simulated using **dnscat2**). The graph shows that PCapper was able to flag all 10 malicious packets but at the cost of 27 benign or safe packets.

DNS Tunneling Attack Detection Algorithm Visualization:



The graph above demonstrates how the algorithm works when processing DNS queries. In this example, 5 queries are processed, once one query meets one of requirements of a potential tunneling attack, it gets flagged. To reduce false positives, the algorithm makes sure that a query has been flagged at least twice before marking it as a malicious or suspicious query or packet.

Conclusions

PCapper met its objectives of detecting DNS, HTTP, and SSH tunneling activities in real-time using Deep Packet Inspection (DPI). The system demonstrated its effectiveness by identifying high-entropy DNS queries, unauthorized HTTP/SSH requests, and other suspicious traffic patterns without relying on resource intensive machine learning models. These results highlight the advantages of DPI in providing real-time, explainable insights into network behaviour, especially in detecting obfuscated or malicious activities.

Recommendations

To improve PCapper, the following recommendations are given:

- Refinine the tool to support additional protocols like FTP.
- Make it so the tool also blocks suspicious malicious traffic. Making it an anti-virus for network packets.
- Optimize packet parsing and processing algorithms to reduce latency and improve performance in high-traffic networks.
- Add real-time notification to enhance usability.

By implementing these recommendations, PCapper can evolve into a great solution for real-time network security, addressing modern challenges in encrypted traffic analysis and tunneling detection.

References

Deri, L., & Fusco, F. (2021). Using Deep Packet Inspection in CyberTraffic Analysis. *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, 89–94. https://doi.org/10.1109/CSR51186.2021.9527976

Dizdar, A. (2023, July 25). DNS Tunneling: How it Works, Detection and Prevention. *Bright Security*. Retrieved October 28, 2024, from https://brightsec.com/blog/dns-tunneling/

Malware-Traffic-Analysis.net. (2019, September 25). Data dump: Trickbot infection, gtag ono19. Retrieved November 1, 2024, from https://www.malware-traffic-analysis.net/2019/09/25/