Grenze International Journal of Engineering and Technology, June Issue

Pattern-based Stateful DPI Intrusion Prevention Engine

Krishnapriya S[1] and Daniel Madan Raja S[2]
[1]Undergraduate Researcher, Department of Computer Science and Engineering, Karunya Institute of Technology and Sciences, Coimbatore, TamilNadu-641114, India
Email: krishnapriyas@karunya.edu.in
[2]Associate Professor, Department of Computer Science and Engineering, Karunya Institute of Technology and Sciences, Coimbatore, TamilNadu-641114, India
Email: danielmadanrajas@karunya.edu

***Abstract*— It has become a huge challenge to ensure security in real time due to the rapid growth of encrypted and high-speed networks. Traditional intrusion detection systems often depend on offline analysis or signature databases, hence are less effective against live threats. This work proposes the design and implementation of a pattern-based stateful Deep Packet Inspection engine for real-time intrusion prevention. The developed system captures live network traffic, inspects packet headers and payloads, classifies traffic based on rule-based analysis, and blocks policy-violating domains using a dynamic blocklist. The proposed engine allows DNS and TLS traffic inspection and raises real-time alerts against malicious or unauthorized communication. The experimental results show that suspicious traffic can be detected and blocked efficiently with low processing overhead, making the solution suitable for use in an enterprise and smart infrastructure network.**

***Key Words*— Deep Packet Inspection, Intrusion Prevention System, Network Security, Stateful Inspection, DNS Filtering, TLS Traffic Analysis.**

I. INTRODUCTION

The more the world is getting dependent on internet-based services and cloud infrastructures, the more network security has become crucial. A modern network has a continuous threat of malicious communications from malware, unauthorized access, and data exfiltration. Intrusion Prevention Systems play a major role in detecting and blocking malicious traffic in real time [10][16].

While DPI facilitates in-depth analysis of network packets beyond simple header inspection by examining protocol information and payload patterns [9]. However, the increasing use of encrypted protocols such as Transport Layer Security (TLS) introduces challenges for effective traffic inspection [6].

This paper discusses the design of a pattern-based stateful DPI engine that is capable of real-time analysis of network traffic, domain-based blocking, and generation of alerts. The system is designed to be practically deployed without using any machine learning technique, thus making it lightweight, explainable, and suitable for real-world environments.

II. RELATED WORK

Over the last 20 years, there has been significant research on network intrusion detection and prevention systems [9][16].

Grenze ID: 01.GIJET.12.2.440_24

Historically, the majority of these systems have been based on signatures that match the signatures of known attacks to the packets seen on a network [3][19]. Some very popular examples of this technology are Snort and Suricata [3][19]. These systems use a large number of rules to detect and prevent attacks and can be very effective when correctly configured with the latest signature files [3]. Unfortunately, configuring these systems can become quite complex, require constant updating of signature files, and, as such, can create a heavy maintenance burden and reduce the ability of these systems to respond to continually evolving threats [9][16].

Recent advancements in research have allowed the development of machine learning and deep learning models to classify networks and detect anomalous traffic [1][18]. These new approaches are designed to identify unknown attacks based on the patterns learned from large training datasets [1]. While current machine-learning methodologies are encouraging, they face challenges concerning the availability of training datasets, complexity of training models, the interpretability of trained models, and deployment to identify attacks in real time [9].

We are developing a lightweight, rule-based, stateful DPI engine that focuses on the inspection of DNS and TLS traffic [4][[5][6]. In doing so, we are avoiding the complexity commonly found in ML implementations and have instead created an effective real-time Intrusion Prevention System for today's encrypted (TLS) networks [6][10][16].

III. System Architecture

This paper presents a new Intrusion Prevention Engine that uses "Pattern-Based Stateful DPI" technology to monitor, analyze and control live network data (traffic) in real-time [9][10]. The architecture is designed to be "modular" (i.e., each module performs a specific function) and "layered" (i.e., each layer provides a level of protection to an entire system) allowing for scalability, maintainability and a minimal amount of processing overhead [16]. Each of the modules in the proposed engine will perform the specific functions required to inspect and detect malicious and/or policy-violating network activity using pattern recognition.

Figure 1 shows the overall architecture of the proposed Pattern-Based Stateful DPI Intrusion Prevention Engine.

Traffic flows in and out of a network interface layer (the layer that receives the live traffic from the host's active network interfaces, e.g. Wi-Fi, Ethernet) [20]. The Live Packet Capture (LPC) module will have access to both inbound and outbound packets, which allows it to detect both purposefully generated traffic (e.g. users) as well as background traffic (e.g. systems).

Capturing packets at this level of the stack provides the DPI engine with a method to see both ends of the communication [9] and allows it to function independently of application-level control or filtering.

When packets are captured, the packet capture module is responsible for sending them to inspection. The packet capture module will collect and analyze packets based on what it sees being sent over the internet. The packets captured will then be forwarded on to the protocol identification module for analysis of the packet header to determine which protocol is being used by the packet.

The protocol parser has specific protocol parsers dedicated to handling DNS and TLS traffic [4][5][6]. The DNS parser will provide an easy way to extract information from query and response packets regarding the domain name [4][5], which can be utilized for domain filtering and policy application. The TLS parser only identifies that traffic is encrypted and provides the application with the relevant metadata associated with that traffic [6], without having to attempt to decrypt the traffic (ensuring compliance with security and privacy laws).

Any traffic that does not correspond to one of the supported protocols is sent to a Generic Packet Handler, providing the system with robustness. The stateful DPI engine maintains context across several packets [9][10]. It also maintains a continuous sequence of packets and uses that sequence to aid in identifying suspicious activities. The stateful DPI engine looks at the sequence of each packet and can therefore detect repeated violations of policy and abnormal traffic patterns. The core of the system is the classification engine that classifies traffic through predetermined rules according to three categories: safe, blocked, or malicious. DNS packets are checked against a configurable blocklist of restricted domains [10], while payload inspection looks for known malicious signs.

TLS traffic analysis is done based on metadata and protocol specifics. Rule-based classification is transparent and easy to set up, and has a low computational requirement. Traffic classified as safe is allowed to proceed without interruption, ensuring normal network operation. Traffic identified as blocked or malicious is forwarded to the blocklist enforcement module, which triggers prevention mechanisms. This module ensures that unauthorized or suspicious communication is immediately restricted, preventing further interaction with the destination host. The alert generation module is activated upon detection of blocked or malicious traffic. It generates real-time alerts displayed on the system console and records detailed logs containing timestamps, protocol information, and classification results.

These logs support forensic analysis, auditing, and result evaluation [10][16]. Separate log files are maintained for general traffic, blocked events, and alerts, enabling structured data analysis. Overall, the proposed architecture provides an efficient and practical solution for real-time intrusion prevention in modern encrypted networks. Its modular design allows easy extension to additional protocols and integration with monitoring dashboards. By combining stateful inspection with rule-based classification, the system achieves effective security enforcement while maintaining low latency and operational stability.

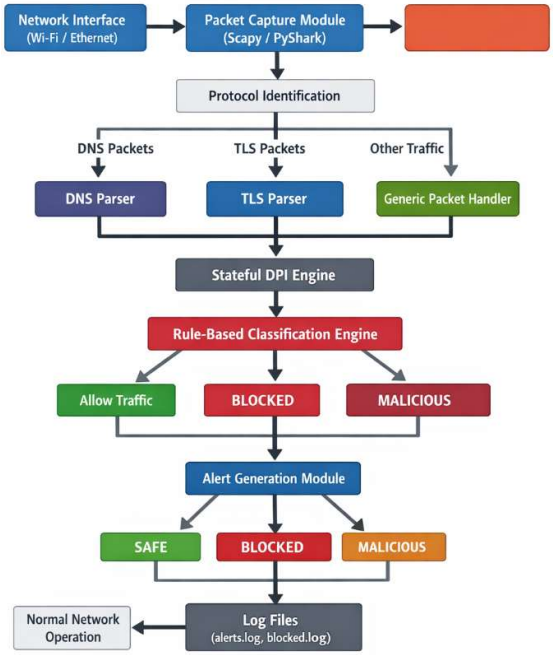

Figure 1: Architecture Diagram

Figure 1 shows the overall architecture of the proposed DPI engine.

IV. Methodology

Using a technique called Deep Packet Inspection (DPI) with Stateful Inspection, the proposed IPS continuously collects live traffic from your computer's network card [9][10][16]. By processing each captured packet in the order that they arrive at the IPS (instead of individually) [9], this system keeps track of the connection so that it can detect violations of policy and/or other image abnormalities in multiple packets – instead of just checking a single packet.

The first step in processing captured packets involves determining what type of protocol the packet is using by analyzing the protocol headers [9][20]. Once the type of protocol has been identified, the captured packet is sent to its associated protocol parser (i.e., captured DNS packets are processed to extract DNS Domain Name info from the queries and responses). Extracting DNS Domain Name info is important because it can be used for Domain-Based Content Filtering and Policy Enforcement [4][5]. Once the packets have been identified and classified, they're processed using their respective protocols. For example, when the system identifies packets as being associated with the TLS (Transport Layer Security) protocol [6], the packets will be treated as encrypted data by the system without being decrypted to ensure that the privacy is sufficiently protected [6][11].

Once the protocol parsing has been completed, the information contained in the extracted packets will be sent to the classification engine. The classification process is rule based and deterministic [10][16], so anyone reading the resulting decisions regarding security policy will find the decision-making process to be easy to interpret and alter as necessary.

DNS domains are checked against a predefined blocklist of blocked or otherwise unauthorized domains [10]. The payload can also be inspected for any known malware patterns using techniques such as keyword detection [18].

Once classifying is complete, the resulting packets will be classified as "safe," "blocked," or "malicious." Safe packets are allowed to continue through the network without any interference, thereby maintaining normal operation. Blocked or malicious packets will cause an alert to be generated and recorded at the same time, along with the appropriate packet detail and timestamping information. This type of architecture requires less processing time and has lower latency than other methods of Intrusion Detection System (IDS) [9][16].

V. IMPLEMENTATION

It is suggested to use the proposed Deep Packet Inspection engine with the use of the “Python” language, considering its simplicity and clear structure, as well as the presence of enough libraries supporting the development of problem-solving tools for network inspection and security issues [7][8]. It should be highlighted that the use of the “Python” programming tool allows for the development of such an inspection tool with acceptable performance during real-time inspection due to the modular structure of the suggested tool [10][16].
The system runs in an active live network environment [9], capturing live real-time information based on the types of protocols identified and defining the various alerts based on these types.

A. Packet Capture Mechanism

To achieve the task of capturing the network traffic, the system relies on the low-level approach of "packet sniffing" with the aid of Python-based "packet capture" libraries [7][8]. This enables the DPI engine to directly monitor the packets from the interface of the network. This can be accomplished without making any significant modifications in the prevailing network infrastructure.
The packet capture module listens for both incoming and outgoing packets and only forwards the captured packets to the further stages of processing. The system in this case minimizes performance overhead. The packets are filtered to ensure that only relevant packets are captured. The system ensures that both background operating system-generated packets and the user-generated packets are captured.

B. Modular Design Architecture

A modular design methodology was used [16] in order to split the DPI engine into well-defined, distinct functional components:

- Packet Capture
- Parsing Protocol
- Classification Engine
- Alerting and Logging

Each module works autonomously and communicates by means of well-defined data formats. This allows updating and extending the individual modules without having an impact on the very core of the system, while new protocol parsers and detection rules can be easily added [10][16] with only slight modifications of the current architecture.

C. Protocol Parsing Modules

Two protocol-specific parsing modules were implemented to analyze DNS and TLS traffic [4][5][6], as these protocols are commonly used and are critical from a security monitoring perspective.

DNS Parsing Module

The DNS parser extracts both DNS queries and DNS responses from captured packets. On the one hand, the domain name of the DNS query that the client requests is extracted; on the other hand, the resolved domain information in the DNS response is recorded. These extracted domain names will be used as the main input of the domain-based filtering.
DNS traffic is also one of the most valuable sources of network intelligence [4][5][9] in cybersecurity because most malicious communications start with a DNS lookup. Therefore, the analysis of DNS packets allows for the detection and prohibition of unauthorized or restricted access to any app well in advance.

TLS Parsing Module

The TLS parser identifies encrypted Transport Layer Security protocol traffic. While the transport layer on the request payload is encrypted, the system is able to identify the TLS sessions and, at times, extract information such as the Server Name Indication (SNI) [6] from them. This information proves useful in obtaining the domain name.
The system thus maintains the encryption boundary [6][11] while focusing on the metadata instead of the payload decryption.

D. Blocklist Management

The blocklist is represented as a simple text-based file [10] that represents a set of domain names that are restricted. This design choice was guided by the need to ensure that the system was highly flexible. System administrators can alter the blocklist dynamically based on their specific organizational requirements without having to alter the DPI engine design in any manner.

During the runtime, the blocklist is loaded into memory, which is later compared with the extracted DNS queries as well as the TLS. If a match exists, the concerned traffic is marked as blocked.

This distinction between policies and logic promotes better security [16], simplicity, and best practice like that found in real-life intrusion prevention systems.

E. Classification Engine

The classification engine is essentially the core decision-making body of the DPI system [10][16] . All the packets of data received from the packet parser are analyzed. The packets of data are analyzed using a set of specified rules, and the decision is made on how they can either be classified as

- Safe
- Blocked
- Malicious

These classification rules include:

- Domain-Based Blocking Using the BlockList
- Detection of suspicious keywords in the packet payloads
- Identification of encrypted traffic patterns

A system like this, which uses rules, has a determinism that gives it transparency [16], thus making analysis simpler, especially in a research scenario.

F. Alerting and Logging Mechanism

Whenever traffic is classified as either Blocked or Malicious, the alerting module can also generate a real-time alert with a timestamp and packet details [10]. Events are logged in structured format to enable later analysis and auditing.

The logging mechanism reinforces forensic investigation by enabling clear logging of [9][10]:

- Blocked domain access attempts
- Suspicious activity detected
- Normal encrypted traffic patterns

All these logs can be used in a higher plane for the study of traffic behavior, refinement of detection rules, and effectiveness enhancement of the DPI engine.

G. System Testing and Deployment

The system has also undergone a test in a normal operating environment with a network [9]. In the test, the normal web surfing traffic and the normal operating system communication have been captured. The test case is designed to ensure that the engine can operate.

As can be seen from the results, the system can effectively detect blocked domains, identify whether the network is being used to send encrypted traffic, and alert accordingly while functioning normally.

VI. RESULT

The proposed Deep Packet Inspection (DPI) engine was successfully implemented and tested in a real network environment using a Windows-based system. The system was evaluated by capturing live network traffic generated through normal web browsing and background operating system services. The evaluation focused on validating the accuracy of protocol identification, effectiveness of domain-based blocking, and reliability of the alert and logging mechanism.

During execution, the DPI engine was able to correctly identify and classify different types of network traffic, including DNS queries, DNS responses, and TLS-encrypted traffic. DNS packets were parsed in real time to extract queried domain names and corresponding responses. When a domain listed in the blocklist file was detected, the system immediately flagged the traffic and generated an alert indicating that the request was blocked. This behavior was consistently observed when accessing blocked domains such as social media and restricted websites.

TLS traffic, which represents encrypted HTTPS communication, was also successfully detected and classified. Although the payload of TLS packets remains encrypted, the system reliably identified TLS sessions and categorized them as safe when no blocklist or malicious indicators were present. This demonstrates the system's ability to function effectively even in modern encrypted network environments where payload inspection is limited.

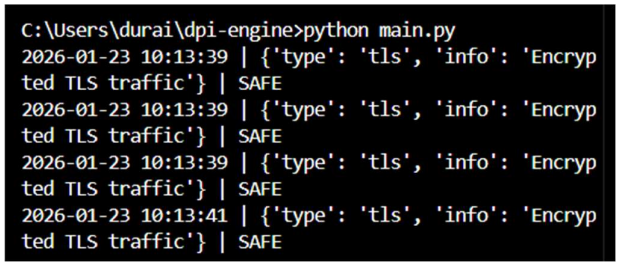

Figure 2: SAFE traffic detection

Figure 2 illustrates safe traffic detection during normal network operation.

Figure 3: Blocked domain detection

Figure 3 shows blocked domain detection when restricted domains are accessed.

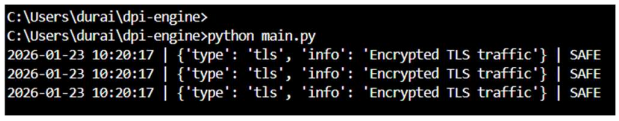

Figure 4: Continuous monitoring proof

Figure 4 demonstrates continuous monitoring of live network traffic.

VII. Conclusion

This paper is about a kind of security system for networks. It is called a Deep Packet Inspection Intrusion Prevention Engine. It uses patterns to keep the network safe. The system looks at the traffic on the network, including DNS and TLS traffic without having to decrypt it. This means it can keep the network secure without invading peoples privacy.

The people who made the system tested it and found out that it can identify protocols accurately and block traffic from certain domains. It can also send out alerts when something bad happens. It does all of this without using up too much of the networks resources. The system is not too heavy and it uses rules so it is a good choice for big companies and smart cities that have a lot of network infrastructure [10][16]. The Deep Packet Inspection Intrusion Prevention Engine is an option, for these places because it is lightweight and easy to use. Future work may include support for additional protocols and integration with centralized monitoring dashboards.

References

[1] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," IEEE Symposium on Security and Privacy, pp. 305–316, 2010.

[2] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of P2P traffic using application signatures," Proceedings of the World Wide Web Conference (WWW), pp. 512–521, 2004.

[3] M. Roesch, "Snort: Lightweight intrusion detection for networks," USENIX Conference on System Administration, pp. 229–238, 1999.

[4] P. Mockapetris, “Domain names—Concepts and facilities,” RFC 1034, Internet Engineering Task Force (IETF), 1987.
[5] P. Mockapetris, “Domain names—Implementation and specification,” RFC 1035, Internet Engineering Task Force (IETF), 1987.
[6] E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.3,” RFC 8446, Internet Engineering Task Force (IETF), 2018.
[7] G. Combs et al., “Wireshark: Network protocol analyzer,” Wireshark Foundation, 2023. [Online]. Available: https://www.wireshark.org
[8] P. Biondi, “Scapy: Packet manipulation tool for Python,” INRIA, France, 2018. [Online]. Available: https://scapy.net
[9] S. Axelsson, “Intrusion detection systems: A survey and taxonomy,” Technical Report, Chalmers University of Technology, Sweden, 2000.
[10] K. Scarfone and P. Mell, “Guide to intrusion detection and prevention systems,” NIST Special Publication 800-94, 2007.
[11] S. Kent and K. Seo, “Security architecture for the Internet Protocol,” RFC 4301, Internet Engineering Task Force (IETF), 2005.
[12] R. Perlman, “Network layer security: Past, present, and future,” IEEE Security & Privacy, vol. 8, no. 1, pp. 58–61, 2010.
[13] A. Moore and K. Papagiannaki, “Toward the accurate identification of network applications,” Passive and Active Measurement Conference, pp. 41–54, 2005.
[14] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, “BLINC: Multilevel traffic classification in the dark,” ACM SIGCOMM Computer Communication Review, vol. 35, no. 4, pp. 229–240, 2005.
[15] N. Feamster, J. Rexford, and E. Zegura, “The road to SDN: An intellectual history of programmable networks,” ACM SIGCOMM CCR, vol. 44, no. 2, pp. 87–98, 2014.
[16] K. Scarfone, P. Mell, “An overview of intrusion detection and prevention systems,” IEEE Security & Privacy, vol. 7, no. 1, pp. 56–63, 2009.
[17] J. Anderson, “Computer security threat monitoring and surveillance,” Technical Report, James P. Anderson Company, 1980.
[18] A. Khurana, M. Gupta, and M. Singh, “Network intrusion detection system using rule-based techniques,” International Journal of Computer Applications, vol. 120, no. 1, pp. 1–6, 2015.
[19] V. Paxson, “Bro: A system for detecting network intruders in real-time,” Computer Networks, vol. 31, no. 23–24, pp. 2435–2463, 1999.
[20] IEEE Standards Association, “IEEE 802.3 Ethernet Working Group,” IEEE, 2022.