MALICIOUS WEBSITE DETECTION AND BLOCKING SYSTEM USING eBPF

PROJECT SYNOPSIS

OF MAJOR PROJECT

BACHELOR OF TECHNOLOGY

COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY

RAVINDER KAUR	RANBIR SINGH	AMRINDER SINGH
2104168	2104166	2104066

JANUARY 2025

UNDER THE GUIDANCE OF
ER. KAMALJEET KAUR

GURU NANAK DEV ENGINEERING COLLEGE, GILL ROAD, GILL PARK, LUDHIANA

OBJECTIVES

1. To detect websites based on Deep Packet Inspection.
2. To log the website traffic to a database.
3. To block websites using eBPF.

LITERATURE REVIEW

1. Encrypted Traffic and Challenges in Classification

   (D. Shamsimukhametov, A. Kurapov, M. Liubogoshchev and E. Khorov, 2024)

   The evolution of encryption protocols, particularly Transport Layer Security (TLS), has significantly enhanced privacy but has also introduced challenges in traffic classification. The Encrypted ClientHello (ECH) extension to TLS, aimed at concealing sensitive metadata like the Server Name Indication (SNI), exacerbates these challenges by rendering key classification features unavailable. While ECH enhances privacy, it necessitates the development of innovative methods that can utilize alternative features for TC.

2. Role of Metadata in Traffic Classification

   (R. Mohite and B. Thangaraju , Sydney, Australia, 2024)

   Despite the increasing adoption of ECH, some non-random TLS metadata remains accessible due to protocol limitations and backward compatibility. These unencrypted metadata elements, coupled with statistical features of traffic flows, provide an opportunity to design classifiers that can perform early TC even in highly encrypted environments. Prior studies have leveraged statistical analysis of network traffic flows, but these methods often suffer from reduced accuracy when applied to encrypted traffic.

3. Packet Filtering and eBPF

   (D. Scholz, D. Raumer, P. Emmerich, A. Kurtz, K. Lesiak and G. Carle, Vienna, Austria, 2018)

   Packet filtering plays a pivotal role in network security, especially in preventing malicious traffic from exiting or entering containers. Traditional packet filtering mechanisms, such as iptables, while widely used, may not be efficient enough to handle the dynamic nature of modern containerized environments. The growing complexity of container orchestration platforms introduces new challenges in maintaining a security perimeter and ensuring traffic flows are properly secured.

METHODOLOGY

eBPF (Extended Berkeley Packet Filter): In the context of **Deep Packet Inspection (DPI)**. eBPF is a powerful technology that allows for safe and efficient execution of user-defined programs in the kernel space without modifying the kernel code. This is particularly useful for tasks like network monitoring, security, and DPI.

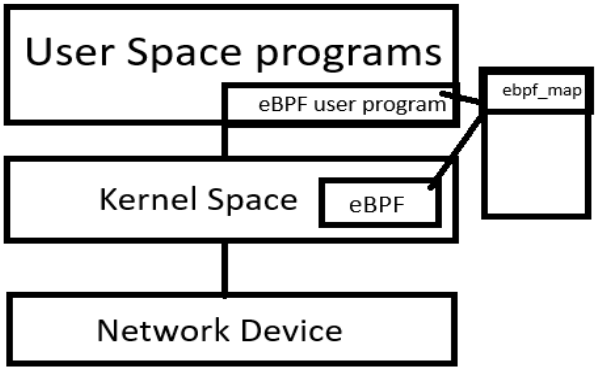


Diagram Components:

1. **User Space Programs:**

    - These are the applications running in the user space, which interact with the kernel through system calls.
    - **eBPF User Program:** A program written in C (or compiled from higher-level languages) that defines the logic for packet inspection. This program is loaded into the kernel using tools like bpftool or libraries like libbpf.

2. **Kernel Space:**

- The core part of the operating system that has direct access to hardware and system resources.
- **eBPF:** This is where the eBPF program is executed. It runs in a sandboxed environment within the kernel to safely process packets received from the network device.
- The eBPF program can hook into various kernel events (like network packets, system calls, etc.) to inspect and filter traffic in real time.

3. **Network Device:**

- Represents the physical or virtual network interface receiving or sending packets.
- The packets from the network device pass through the kernel where the eBPF program can inspect them before they're delivered to user space or other parts of the kernel.

4. **eBPF Map:**

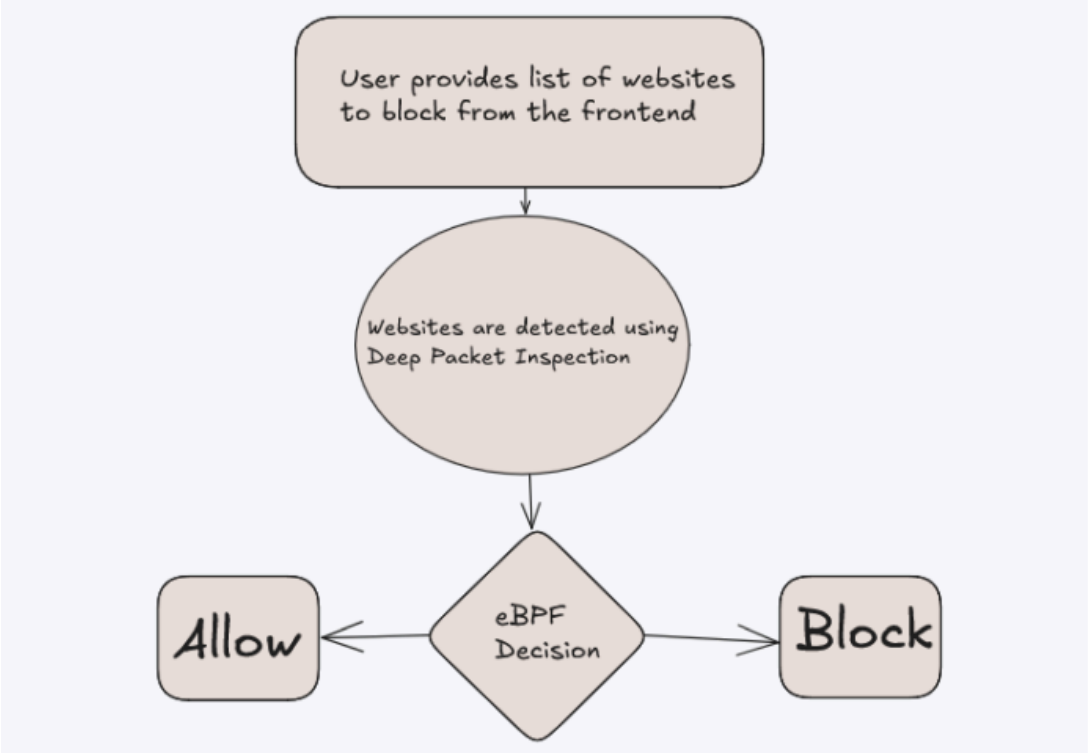
- A shared data structure that allows communication between user space programs and eBPF code in the kernel.
- **Purpose:**

    - Store packet metadata, statistics, or DPI-related information.
    - Enables real-time decision-making by updating policies without unloading the eBPF program.

Architecture of DPI:

1. **Packet Capture:** Network packets arrive at the network device.
2. **Packet Processing:** The kernel passes these packets through the eBPF program.
3. **Inspection:**

    - The eBPF program analyzes packet headers and payloads for DPI purposes, such as detecting anomalies, malicious patterns, or specific protocol data.
    - The program uses **eBPF maps** to store or retrieve data dynamically.



4. **User Space Interaction:** Results from the inspection (like suspicious activity alerts) are sent to user space programs for logging, further analysis, or action.