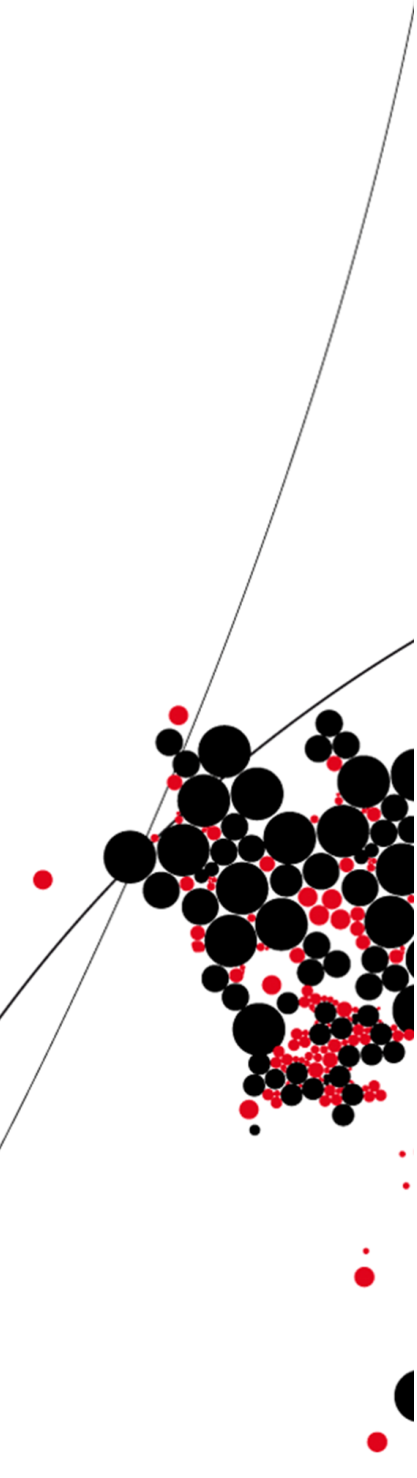
UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science

A framework for detecting and preventing DoS attacks in automotive Ethernet switches

Henil Goswami

M.Sc. Thesis
March 2023



Examination Committee:
prof. dr. ir. G. J. Heijenk, UT
dr. A. Sperotto, UT
ir. E. Molenkamp, UT
dr. Rajeev Roy, NXP

Design and Analysis of Communication Systems Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Acknowledgement

I take this opportunity to express my sincere thanks to everyone who supported me during my thesis . First and foremost, I would like to express my gratitude to my supervisor, Dr. Rajeev Roy, for his guidance and support throughout my research. His patience, expertise, and encouragement to explore challenging topics have been invaluable in improving the quality of my thesis.

I would also like to extend my sincere thanks to my university supervisor, Prof. Dr. Ir. Geert Heijenk, for his insightful comments and suggestions which helped me navigate the research process and improve my work.

My gratitude is also extended to my examination committee, Dr. Anna Sperotto and Ir. Bert Molenkamp, for their valuable time and effort in reviewing and assessing my work. Their feedback and suggestions have been helpful in shaping the final outcome of my research.

Finally, I would like to express my gratitude to my parents and friends for their unwavering love and support throughout my academic journey. Their constant encouragement and motivation have been my driving force.

Abstract

Distributed denial of service (D)DoS attacks on in-vehicle networks (IVNs) can disrupt the operation of the vehicle and pose a risk to passenger safety and the safety of other road users. These attacks can be challenging to detect and prevent because they often involve a large number of devices generating traffic simultaneously, making it hard to distinguish malicious traffic from normal traffic. In this research, we aim to develop and evaluate a (D)DoS detection and prevention system for Ethernet-based in-vehicle networks. Ethernet switches and gateways, which are commonly used in these types of networks to forward and process traffic, could potentially be accessed by hackers to compromise the network. Our proposed approach combines entropy-based (D)DoS detection with access control list (ACL) based traffic filtering to detect and mitigate attacks. The results showed that the entropy-based (D)DoS detection method demonstrated excellent performance, with a high detection rate and a low count of false negatives in detecting attacks. However, there was a limitation in identifying certain types of attacks that exhibit similar patterns to legitimate traffic. Evaluation for ACL-based traffic filtering showed that the created ACL rule effectively filters out (D)DoS traffic, but it also dropped legitimate packets with the same header fields as (D)DoS packets. By developing and evaluating a systematic approach for protecting automotive Ethernet networks against this network attacks, we aim to improve the security of IVNs.

Contents

Chapter 1

Introduction

Many electronic control units (ECUs) are employed in today's vehicles to improve the overall vehicle ecosystem's performance, security, and safety. ECUs typically contain over 100 million lines of code for complex functions, which is 95 times more than in commercial aircraft [1]. As the automotive industry trends toward connected and autonomous vehicles, security in complex communication across vehicle networks has become challenging. The increasing demand for connecting vehicles requires automakers to create sophisticated software and electronics designs to meet connective needs [2][3]. McKinsey & Company estimates that the market share of connected vehicles will continue to rise from the current 50% of the retail connected vehicles to 95% worldwide sales in 2030 [4].

Introducing data-intensive systems such as infotainment systems and advanced driver-assistance systems (ADAS) necessitates the adaptation of in-vehicle networks. Traditional bus technologies such as CAN, MOST, FlexRay, and LIN are not future-proof when it comes to high-speed data transfer and bandwidth requirements (maximum bandwidth of 1 Mbps for CAN and 10 Mbps for FlexRay). Consequently, Ethernet technologies are used to establish efficient and straightforward communication in modern and future-generation vehicle networks. In addition, vehicular networks extend to vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), and infrastructure-to-vehicle (I2V) networking. The change in network architecture and the introduction of Ethernet as a communication backbone necessitates to research and develop network security in vehicles.

One of the potential security threats is the (D)DoS attack, which can disrupt the operation of the vehicle. In the context of in-vehicle networks, a DoS attack can occur in two ways. For example, a hacker could create a command that stops the engine of the car, preventing the passenger from using it. However, this type of attack requires deeper understanding of the network and the ability to gain control. The other form of (D)DoS attack is relatively simple, as it seeks to deplete the victim's resources by flooding it with a high volume of malicious traffic. This research focuses on the latter type of (D)DoS attack, which aims to exhaust the bandwidth rather than command and control attacks.

1.1 Challenges

Connected vehicles and autonomous cars are two significant trends adopted by today's automotive industries. The common goal of this trend is to significantly reduce accidents and make the driving experience less tedious. As per the WHO, there are 1.3 million deaths caused by road accidents every year involving human errors like speeding and distracted driving [5]. Vehicles need to be much more intelligent and trustworthy to reduce road accidents. Achieving such goals requires high-speed data communication and sensors. Moreover, with the expectation of connected vehicles and access to the cloud infrastructure, security becomes a significant challenge.

The major cybersecurity challenge was first experienced by General Motors (GM) vehicles on their On-star system in 2010 [6]. This attack allowed hackers to unlock and start the car remotely by exploiting vulnerabilities in the onboard computer. The On-Star entertainment system in GM automobiles was exploited in an attack that affected several million vehicles. To address the issue, the company released an update that could be installed wirelessly to protect against the hack. It took GM approximately five years to completely protect this vulnerability [6]. Similarly, in 2015, Jeep vehicles experienced a cybersecurity attack on their U-connect system. The vulnerability allowed hackers to remotely control the vehicle through its IP address. It includes controlling the brakes, windshield wipers and turning on and off the engine. Fiat Chrysler Automotive (FCA) had to recall 1.4 million vehicles due to the vulnerability in its systems. To protect from the attack, FCA distributed the updates through USB drives to the vulnerable cars [7].

Other significant attacks include the Tesla hack and the attack on Volkswagen keys [8], [9]. The vulnerability in Tesla was observed in 2016 due to a software issue. The attack allowed hackers to access the door open/close command and brake it. The software issues were resolved by over-the-air updates [8]. On the other hand, a vulnerability in Volkswagen vehicles was detected due to a hardware issue. The hacker exploited the vulnerabilities in the transponders in the key. Due to the lack of encryption methods between the transponder in the key and the receiver in the car, a hacker can steal a car. The entire hardware of the lock system was required to be changed to solve the issue [9].

There are several components in the car that could compromise cyber security. For instance, an in-vehicular network can be hacked in two primary ways; physically, where the hacker can physically access the vehicle, and over the air, where the hacker can remotely access the vehicle. Physical hacking involves accessing an in-vehicular network with open USB ports or onboard diagnostic (OBD) ports. In comparison, over-the-air involves devices connected to the internet like the infotainment system, downloading third-party applications, or other devices like Bluetooth-based tire pressure monitoring systems or wireless car access keys.

1.2 Problem Statement

Modern vehicles are equipped with various systems, such as advanced driving assistance systems (ADAS), telematic control units (TCU), and in-vehicle infotainment (IVI) systems that interact with the in-vehicle network and the outside world. These systems are typically connected with network switches and gateways to forward and process network traffic. A hacker could gain access to the network gateways or switches to compromise the security of the in-vehicle network. Modern Ethernet switches include an access control list (ACL) and rate limiters to process network traffic. The advantage of identifying and filtering malicious traffic on the Ethernet switch is that it offloads the network processing tasks carried out by the gateways or routers. For instance, when a hacker sends a large number of attack packets to the device, the CPU usage of the device increases, and the performance degrades [10]. These attacks are typically mitigated by using an ACL to block the hackers' arrival.

(D)DoS attacks on vehicle networks can disrupt the operation of the vehicle and pose a risk to the safety of passengers and other road users. These attacks can be difficult to detect and prevent, as they often involve a large number of devices generating traffic simultaneously, making it difficult to distinguish malicious traffic from normal traffic. There is a need for a reliable and effective (D)DoS detection and prevention system that can protect against these types of attacks in the automotive Ethernet switch. The aim of this research is to develop and evaluate such a system to improve the security and resilience of Ethernet-based in-vehicle networks.

This problem statement identifies the problem of (D)DoS attacks on automotive Ethernet networks and explains why it is important to identify known attack patterns and define effective ACL rules to protect against these attacks. It also states the research aim of developing and evaluating a systematic approach to this problem and clearly explains the importance of this research in terms of improving the security and reliability of Ethernet-based vehicle networks.

Based on the problem statement, the following research questions (RQ) are formulated and answered in this thesis.

- **RQ 1** What are the potential entry points for a hacker to gain access to an in-vehicle network?

  The aim of this question is to understand how the IVN connects to the external world. By identifying the entry points and components involved, we can better understand the potential vulnerabilities and risks of network attacks. It is relevant to identify and prevent these attacks in order to ensure the security of the vehicle.

- **RQ 2** What are the Ethernet switch's capabilities for mitigating (D)DoS attacks?

The main purpose of answering this question is to gain a better understanding on (D)DoS attacks and the role of Ethernet switches in IVNs. As one of the key components for processing and forwarding network packets is the Ethernet switch, it is relevant to research the capabilities of the Ethernet switch in order to detect and mitigate (D)DoS attacks. By understanding the capabilities of these switches, we can develop a framework for detecting and preventing these types of attacks, which is crucial for maintaining the security of the IVN.

- **RQ 3** How to replicate the Ethernet switch's lookup behaviour in the simulation environment?

  The goal of this question is to design the Ethernet switch's lookup behaviour in a simulation environment. This design will serve as the foundation for our (D)DoS detection and mitigation framework, allowing us to test and evaluate these methods in a controlled and simulated setting. By designing the Ethernet switch pipeline in a simulation environment, we can gain a better understanding of how these methods perform.

- **RQ 4** How to identify known (D)DoS attacks and define ACL rules for Ethernet switches?

  The goal of this research question is to develop a framework for detecting and mitigating (D)DoS attacks. This framework will involve analyzing network traffic to identify (D)DoS attacks, collecting the malicious packets, and creating an ACL rule to block them. The generated ACL rule will be implemented in the Ethernet switch pipeline design in order to effectively mitigate the attack. By providing a systematic approach to detecting and mitigating these attacks, we can improve the security of the IVN.

- **RQ 5** How well does the proposed (D)DoS detection and prevention method perform?

  The aim of this research question is to assess the effectiveness of our proposed (D)DoS detection and prevention algorithm. By evaluating the methodology, we can determine the performance of the algorithm in terms of its ability to accurately detect and prevent (D)DoS attacks. This evaluation will provide valuable insights into the strengths and weaknesses of the algorithm, allowing us to identify areas for improvement and optimize its performance.

1.3 Thesis Outline

The remaining part of the thesis is organized as follows:

Chapter 2 describes background information about the attack surface of IVNs, entities

involved between a hacker and the targeted device, and capabilities of Ethernet switch to detect and mitigate (D)DoS attack. The chapter aims to answer RQ 1 and 2.

Chapter 3 presents the design of an Ethernet switch's lookup capability in a simulation environment. The chapter aims to answer RQ 3.

Chapter 4 presents the methodology to detect and mitigate (D)DoS attacks. Based on the presented method and the design of the Ethernet switch from Chapter 3, we conclude this chapter with a proposed framework. The aim of this chapter is to answer RQ 4.

Chapter 5 presents the evaluation of (D)DoS detection and mitigation method. The chapter aims to answer RQ 5.

Chapter 6 concludes the thesis by providing answers to the research questions and stating future work.

Chapter 2

Background

This chapter is divided into three sections. Section 2.1 discusses the background and state of the art of in-vehicle network (IVN) attack surfaces, including vulnerabilities and potential entry points for attacks. Section 2.2 examines the threat posed by (D)DoS attacks to IVN, including the types of attacks and their consequences. Section 2.3 provides a functional overview of Ethernet switches and their ability to mitigate (D)DoS attacks. The chapter concludes by answering RQ1 (**What are the potential entry points for a hacker to gain access to an in-vehicle network?**) and RQ2 (**What are the Ethernet switch's capabilities for mitigating (D)DoS attacks?**).

2.1 State-of-the-art on Attack surface

To answer RQ1, it is necessary to have in-depth knowledge of the potential entry point to access IVN. Therefore, the goal of this section is to provide an in-depth survey of various IVN components which could be exploited by a hacker. The goal will be achieved by dividing the section into the following parts: In the first part, the surfaces where a hacker can gain access to IVN are discussed in Sub-section 2.1.1. Next, in Sub-section 2.1.2, we will perform a literature study on how a hacker could use these entry points to perform attacks. The section is concluded by concluding remarks in Sub-section 2.1.3.

2.1.1 Attack surface

The entry points for an IVN depend on the attack surfaces that a hacker can access. These surfaces typically include communication channels for connecting to the IVN. In this section, we classify each attack surface into three categories: physical access, short-range wireless access, and long-range wireless access. Figure 2.1 shows the attack surfaces of a connected car. The figure includes three categories based on the position of the hacker and the corresponding attack surface. These categories are long-range wireless access

(A), physical access (B), and short-range wireless access (C). The figure also depicts four potential entry points based on the hacker's position: in-vehicle infotainment, telematics control unit, onboard diagnostic, and sensor interface. We will discuss each of these attack surfaces in detail. In general, a malicious actor may try to send malicious packets using these attack surfaces to carry out network attacks, which could potentially affect the vehicle's non-safety and safety-critical operations.

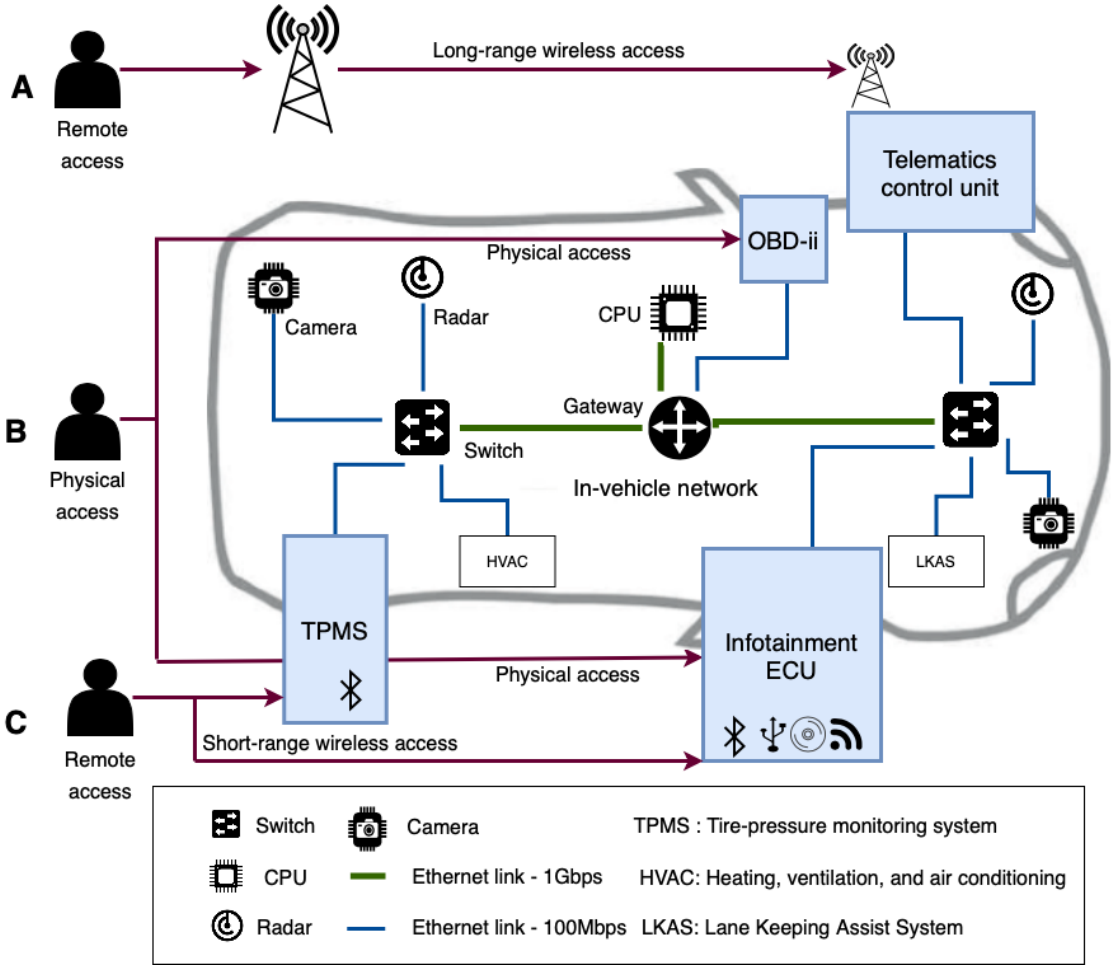


Figure 2.1: An overview of the attack surface with four potential entry points for network access: in-vehicle infotainment, telematics control unit, on-board diagnostic and sensor interface

- **A: Remote access (long-range wireless access)**

  The Telematics control unit (TCU) and in-vehicle infotainment (IVI) systems are two of the devices that serve as a gateway between the cloud and in-vehicle network. These systems typically use long-range wireless communication, such as cellular networks, to connect with the cloud or other V2X standards. The TCU, which collects telemetry data or IVI, which can download 3rd party software over the air, poses a significant risk because it could allow hackers to access IVN from the cloud.

- **B: Physical access**

A hacker must maintain physical access to the vehicle's communication channel to carry out physical attacks. This is done by accessing physical ports like OBD-II, USB, or a malicious node in the IVN. Hackers would typically try to collect data or inject malicious packets to perform attacks on the access node in IVN. Physical attacks may pose fewer security risks than remote access because a hacker needs to maintain physical access.

- **C: Remote access (short-rage wireless access)**

  Systems such as the Tier pressure monitoring system (TPMS) and passive keyless entry system (PKES) use wireless technologies such as Bluetooth, RFID, and Wi-Fi. These wireless technologies have relatively short-range wireless access when compared to cellular networks. A hacker would typically attempt to perform a malicious attack by exploiting the authentication of wireless communication.

2.1.2 Literature review

After knowing the different surfaces where a hacker could perform attacks, it is relevant to know how this attack can be performed. This subsection will elaborate on research which has identified the methods to perform attacks. We use the same classification to categorize attack surfaces to classify methods used by researchers to perform attacks.

- **A: Remote access (long-range wireless access)**

  The advance in the infotainment system and the ability for over-the-air updates require long-range communication. Long-range communication depends on technologies like cellular networks and GPS. GPS is used to position and assist drivers while they are driving. Additionally, fleet management uses it to track the whereabouts of vehicles. Oruganti et al. [11] suggest that a hacker could use a GPS interface as a point of entry to sniff and inject data. Miller et al.[7] obtained GPS data from a vehicle's head unit using an unprotected port connection. Jover et al. [12] have conducted a survey on the impact of (D)DoS attacks on the availability of communication systems in Long-Term Evolution (LTE) cellular networks.

- **B : Physical access**

  The physical attacks involve hackers physically accessing the vehicle. Zhang et al. [13] indicate that hackers can readily collect and analyze car data and control them by injecting messages through the OBD II port. OBD II is a required port to record diagnostic and environmental (such as pollution) data. This interface is directly connected to the in-vehicle network, and by exploiting the vulnerabilities of the OBD port, it is feasible to initiate various attacks that can affect driver safety and security [14]. Checkoway et al. [15] demonstrate various attacks using both wireless OBD dongles and direct connection to an OBD port. Wen et al. [16] indicate

that the hacker can fuzz the diagnostic port of the vehicle through a vulnerable dongle to perform (D)DoS attacks. USB ports are other types of access point that could be vulnerable to in-vehicle networks. USB ports are used in the head units of infotainment systems. Nissim et al. [17] address to override USB firmware which can serve as a (D)DoS attack. Sensors and actuators are used inside vehicles to support a variety of functions, including tire pressure and engine temperature measurement. Loukas et al. [18] have proposed that sensors could be physically attacked by signal jamming to block network traffic between the sensors and the in-vehicle network.

- **C : Remote access (short-rage wireless access)**

  Over-the-air attacks can be further classified as short-range wireless attacks. Wi-Fi and Bluetooth are examples of protocols that are commonly used in short-range wireless access and can potentially expose vulnerabilities in in-vehicle networks. Using scanning tools, Josephlal et al. [19] show that hackers may simply extract the vulnerability of in-vehicle networks. They were successfully able to demonstrate the (D)DoS attack on an infotainment system. Gulliberg et al. [20] demonstrated that hackers can perform (D)DoS attacks on Bluetooth devices by exploiting master-slave communication. According to Francillon et al. [21], hackers can even break into the vehicle by tracking network traffic between the wireless key and the vehicle. Lounis et al. [22] presents the (D)DoS attack on Wi-Fi by exploiting a race condition-based vulnerability. Miller et al. [7] demonstrate how an in-vehicle network is connected to a remote control door lock receiver. Eisenbarth et al.[23] cracked the key fob encryption and performed relay signal attacks to lock and unlock the doors. Rouf et al. [24] show an attack on wireless tire pressure system monitoring (TPSM) that allows the author to conduct eavesdropping operations from a distance of 40 meters from the victim's vehicle. TPSM consists of pressure monitoring sensors that are connected to each tyre to send real-time data to the ECU [7].

2.1.3 Concluding remarks

Modern automobiles include advanced driving assistance systems (ADAS), telematic control units (TCU), and in-vehicle infotainment systems (IVI) that interact with in-vehicle networks and V2X communications. Contradictory to the benefits of these systems, they provide a broad attack surface for hackers to compromise a vehicle's critical systems. Numerous external interfaces exist in vehicles, such as OBD II, which allows for vehicle diagnostics and cellular networks, enabling passengers to connect to the Internet. The security risk of the IVN depends on the location of the hacker and the interconnection of the components that are compromised. We divide hackers' locations based on long-range access, short-range access, and physical access. Based on a literature review, various methods have been identified for performing network attacks on vehicles, including the use of communication channels such as LTE, Wi-Fi, and Bluetooth to carry out (D)DoS attacks.

2.2 Background on (D)DoS attacks

Before addressing to RQ 2 **(What are the Ethernet switch's capabilities for mitigating (D)DoS attacks?**), it is relevant to have a detailed overview of (D)DoS attack and why do we select Ethernet Switch to mitigate (D)DoS attack. This section provides a survey on how (D)DoS attacks can be seen as a threat to automotive networks. Sub-section 2.2.1 lists different types of (D)DoS attacks based on the ICMP, TCP, and UDP protocols. Sub-section 2.2.2 gives the general motive that the hacker may have to perform (D)DoS attacks on vehicles. And Sub-section 2.2.3 lists various stages where (D)DoS attacks can be detected and prevented. We end the section with concluding remarks in Sub-section 2.2.4.

2.2.1 Types of (D)DoS attacks

Ethernet based in-vehicle network is a network that uses the Internet Protocol (IP) to connect the various electronic systems and devices within a vehicle. This type of network enables communication and data transfer between the different systems, such as the ECU, TCU, and IVI.

An Ethernet based in-vehicle network uses the same types of protocols and technologies as a traditional computer network, such as TCP, UDP and ICMP [25]. This allows for the integration of multiple devices and systems, and enables the use of Internet-based services and applications within the vehicle. While this brings benefits, it also exposes the network to potential security threats, as these protocols are commonly leveraged by hackers to launch Distributed Denial of Service (DDoS) and Denial of Service (DoS) attacks.

A DDoS attack utilizes a network of compromised devices to flood the target network with a large amount of traffic, overwhelming the network and rendering it unavailable to legitimate users. As in-vehicle networks are exposed to large external infrastructures, they are particularly vulnerable to DDoS attacks, which can cause disruptions in communication between vehicles and infrastructure. A DoS attack, on the other hand, originates from a single device or a small group of devices, and also disrupts the availability of the network. For example, a hacker could gain access to one of the vulnerable in-vehicle components and initiate a DoS attack on the in-vehicle network.

As ICMP, TCP, and UDP are three of the protocols used in automotive Ethernet, the focus of this subsection will be on (D)DoS attacks carried out by exploiting these protocols. Figure 2.2 depicts the taxonomy of (D)DoS attacks. The (D)DoS attacks can be divided into two parts :

- **Flood attacks**: Flood attacks aim to exhaust the victim's resources with a large number of bogus traffic or request packets. Unlike semantic attacks, flooding attacks

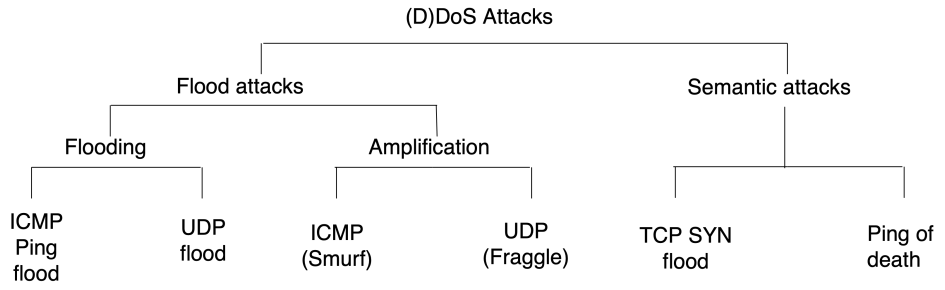


Figure 2.2: Taxonomy of (D)DoS attacks

are much more challenging to mitigate since hacker uses normal-looking packets to perform attacks. This means filtering techniques may also filter legitimate packets, increasing the false negative count. Additionally, the hacker must have more resources compared to the victim to make the attack successful.

- **Semantic attacks**: Instead of exhausting the target's bandwidth, semantic attacks aim to exploit a vulnerability or flaw in the target device. The victim can often reduce the impact of these attacks by using network traffic filtering. Compared to flood attacks, semantic attacks do not need to produce as much traffic as flood attacks.

In the remaining part of this section, we will elaborate on 6 (D)DoS attacks that could be detected by inspecting layer three and layer four headers. Figure 2.2 categorizes each attack vector as either a flood or semantic attack.

- **ICMP ping flood** The Internet Control Message Protocol (ICMP) is a network layer protocol used for applications like ping and traceroute. It is primarily used to assess the connectivity and health of the request's target device. A ping flood attack requires the hacker to know the IP address of the target. The hacker attempts to overwhelm a target device with ICMP echo-request packets, rendering the target device inaccessible to normal traffic. The attack floods the network's incoming and outgoing channels, consuming significant bandwidth. So, the target device is forced to use its resources to respond to each ICMP echo request with an ICMP echo reply packet [26].

- **UDP flood** A UDP flood attack is a (D)DoS attack in which a hacker floods random ports on the targeted device with UDP packets. When a UDP packet is received at a specific port, the target device looks for applications associated with that packet. If the target device cannot find the application, it will return an ICMP destination unreachable packet. As the targeted device continuously receives UDP packets and sends replies for each UDP packet, significant bandwidth is consumed, making the target device unresponsive to other clients [27].

- **ICMP Smurf** An ICMP Smurf attack is a (D)DoS attack that aims to exhaust the target devices with a large number of ICMP echo reply packets. Similar to the ICMP ping flood attack, the ICMP Smurf attack is performed by sending ICMP echo request packets. However, in a Smurf attack, the ICMP echo request is sent to a botnet with a spoofed source IP address of the targeted device. In response to the ICMP echo request by the hacker, the botnet sends an ICMP echo reply to the targeted device from each device present in the botnet. To process all the ICMP echo replies, the targeted device will consume significant bandwidth, which makes the target device unresponsive to other clients [28].

- **UDP Fraggle** The UDP Fraggle attack is a (D)DoS attack and is similar to the Smurf attack. Both of these attacks aim to consume a significant amount of bandwidth on the targeted device by sending a spoofed packet to a botnet. But, in a UDP Fraggle attack, the hacker sends UDP packets, whereas an ICMP Smurf attack involves sending ICMP packets [29].

- **TCP SYN flood** A TCP SYN flood attack uses the three-way handshake to establish a TCP connection. TCP SYN flood attack involves the hacker sending a large number of SYN (synchronize) packets to the target system, which initiates a connection between the attacker and the victim's device. To carry out the attack, the hacker typically sends an SYN packet with a spoofed source IP to the victim's device, and the victim responds with an SYN+ACK packet as a response to the SYN packet. However, the hacker does not reply with the required ACK packet, resulting in an incomplete connection [30].

- **ICMP Ping of death** A Ping of Death is a (D)DoS attack that can bring the system to a halt with a single oversized packet. According to RFC 791, the maximum size of an IP packet allowed is 65,535 bytes, which means that the maximum payload length of an ICMP echo request allowed is 65,507 bytes (65,535 bytes minus the packet header). However, a hacker exploits the payload field to send an ICMP echo request with a payload higher than 65,507 bytes. A device vulnerable to an ICMP Ping of Death attack goes into a DoS condition because the device is unable to reassemble the packet [31].

The above-mentioned (D)DoS attacks are just a few of the many different types that exist. Each attack has its own unique characteristics and can cause varying levels of impact. The ability to detect these attacks by inspecting layer three and layer four headers is crucial for ensuring the security of networks. In the next section, we discus the motivations behind this attacks on vehicles.

2.2.2 Motivations behind (D)DoS attacks on vehicles

After realizing the known (D)DoS attacks, it is relevant to know why someone would hack your car. Principally, a (D)DoS assault aims to stop legitimate users from using a service. In the context of a (D)DoS attack on a vehicle, the hacker aims to prevent the driver from operating the vehicle's functions (e.g., infotainment system) [32]. This section further identifies five reasons why a hacker would hack your car.

- **Financial** The financial motivation of (D)DoS attacks often involves the extortion of the target system. In other words, the target is required to pay a ransom to regain access to their system [33]. If the target is a vehicle, paying a ransom to utilize a car could be a motive. Attacks with a financial motive may result in a loss of client confidence.
- **Political** Hacktivism is primarily driven by political motives. Hacktivism describes attacks where the hacker's primary aim is political expression rather than money. An example of this in the IT industry is the 2015 GitHub attack, which lasted many days [34]. Similarly, in automotive networks, this could be seen as a threat because the general motive for hacktivism is to gather data for political motives.
- **Cyber warfare** In a cyber war, a (D)DoS attack may be used as a weapon to impair the victim's services [35]. The assault on Estonia in 2007 illustrates a (D)DoS attack carried on by cyber warfare [36]. (D)DoS attacks on a car may be seen as a weapon itself. Hijacking cars could be an incentive for cyber warfare.
- **Commercial** It may be viewed that one company will generally benefit if the competitive company is proven to be unreliable and insecure. Executing (D)DoS attacks on the competing company might result in achieving these benefits. Similarly, for car manufacture, one OEM could try to exploit the security of a competing OEM in order to prove the competing OEM's unreliability [37].
- **Emotional** The ease of using services like booter [38] to generate (D)DoS attacks allows people without much technical expertise to generate the (D)DoS attacks. Fewer efforts can allow people to launch (D)DoS attack out of boredom.

2.2.3 Mitigation stage

Depending on the source (remote or physical), hacker traffic involves various stages of packet routes to reach the target device. Henceforth, we will call these stages "mitigation stage". This section aims to provide an overview of the different entities involved between the hacker and the target. We will consider two different scenarios and consider stages at which (D)DoS attacks can be detected. In the first scenario, we will consider TCU as the victim, and in the second scenario, we will consider an in-vehicle component connected

to the TCU via Ethernet switch as the victim. Figure 2.3 represents an overview of the entity involved in routing the packets to the victim. The stages at which (D)DoS traffic can be detected when the victim is the TCU are represented with letters A through D. The stages when the victim is the IVN component (e.g., radar) are represented with numbers I through VI.

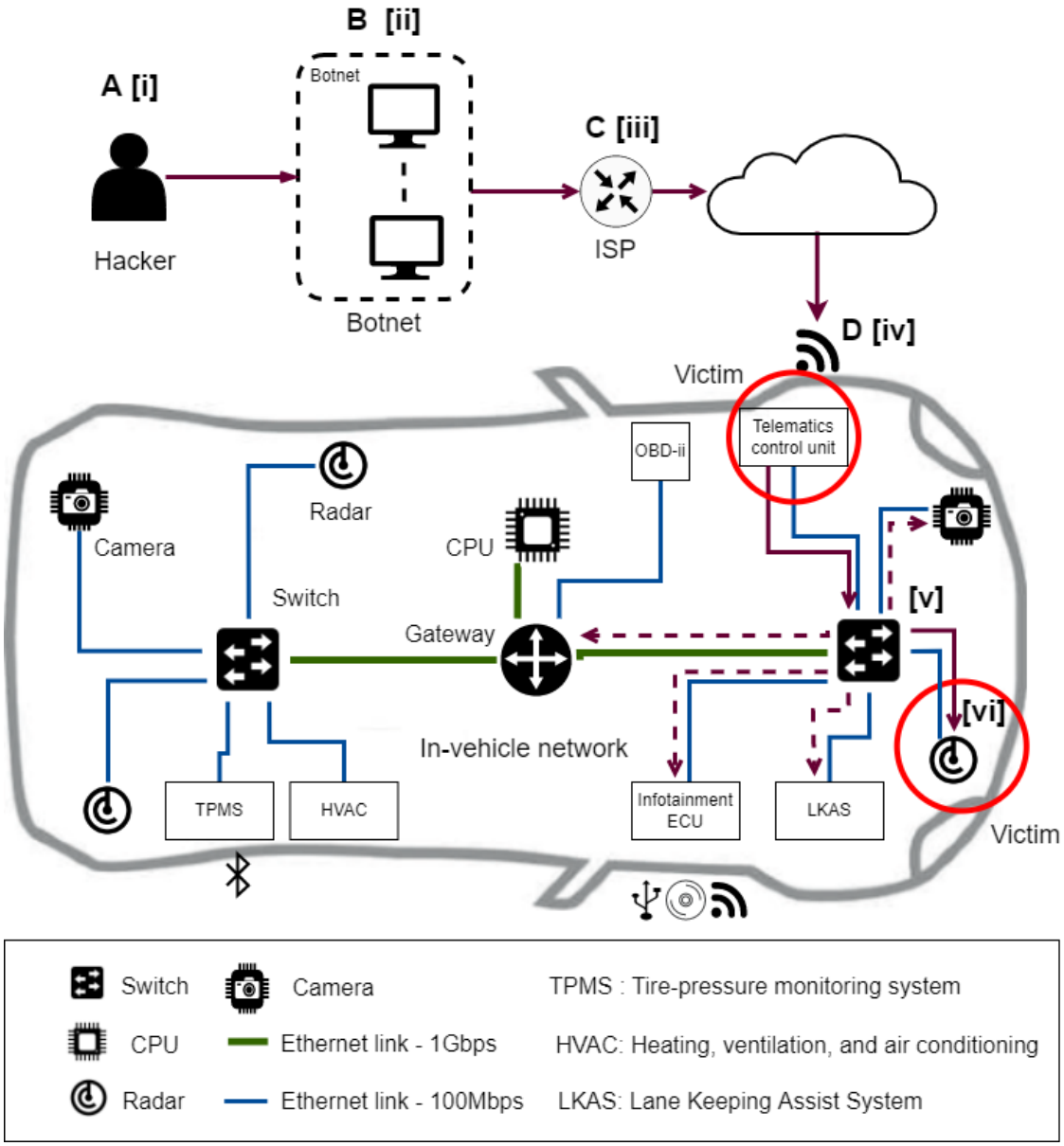


Figure 2.3: Overview of different mitigation locations and attack scenarios marked with A-D (victim: telematics control unit) and i-vi (victim: radar).

- **Hacker stage A [i]**

  (D)DoS assaults can be mitigated at the hacker stage. However, this would be the responsibility of the law enforcement agencies [39]. Mitigation at this stage is outside the scope of this research.

- **Botnet stage B [ii]**

  This stage of mitigation focuses on preventing the use of botnets [40]. This is outside the scope of the study.

- **ISP stage C [iii]**

  This mitigation stage involves deploying an intrusion prevention system by the internet service provider (ISP). Typically, this is not done due to the current net neutrality controversy, which concerns if an ISP may have the authority for traffic filtration based on packet data [41]. Mitigation at this stage is possible even if it is outside the scope of this study.

- **Telematic control unit (Victim) D**

  The telematic control unit is one of the ECUs with access to the cloud via the cellular network. The vehicle uses this interface to interact with other vehicles via V2X standards. In the case where TCU itself is a target, the traffic originating from the cloud could easily cause harm to the TCU as the TCU cannot handle a throughput as large as the ISP's edge router. Implementing an intrusion detection system at this stage allows for greater granularity than at the ISP stage. However, TCU is an aftermarket device, which means they are not under the direct control of the OEM. Mitigation at this stage is possible but depends on the quality of the intrusion detection system implemented by the manufacturer of the TCU.

- **Telematic control unit (Intermediate) [iv]**

  The TCU also serves as a intermediate between the cloud and the IVN. The TCU collects telemetry data from the vehicle, such as speed and engine data, and uses firmware over the air to update ECUs. To enable these functionalities, TCU requires access to the IVN. On the other hand, it poses a significant risk because it allows hackers to access IVN from the cloud. Because IVN is transitioning from a CAN bus to an Ethernet network, there is an end-to-end IP network from the hackers' laptop to the targeted ECU in the vehicle, increasing the risk of an attack against an ECU. As previously stated, mitigation of an attack vector at this stage depends on the quality of the IDPS system implemented by the TCU manufacturer. Similarly, any other ECU that serves as a intermediate between the cloud and IVN may pose a security risk to IVN.

- **Ethernet switch [v]**

  Ethernet-based IVN architectures typically include network switches to forward network traffic to connect ECUs across IVN. Malicious traffic flowing through the switch can be detected and filtered out by performing packet inspection and implementing accesses control list. As mitigation of (D)DoS attacks at the Ethernet switch is the main focus of this research, we will discuss the mitigation in detail in section 2.3.

- **Targeted ECU (Victim) [vi]**

  It is possible to mitigate the (D)DoS attacks on the target itself. Mitigating an attack at this stage requires an IDPS system to be implemented on the target machine. This is outside the scope of the study.

2.2.4 Concluding remarks

The aim of this section was to provide an overview of how a (D)DoS attack could be seen as a threat to IVN. To describe this clearly, we used (D)DoS taxonomy to classify (D)DoS attacks into flood attacks and semantic attacks. We describe six different (D)DoS attack types and realize the motivation hackers could have to perform the (D)DoS attacks on vehicles. We discussed methods to mitigate (D)DoS attacks at different stages. These stages are (I) the hacker, (II) the botnet, (III) ISP, (IV) TCU, (V) the Ethernet switch, and (VI) the targeted ECU. We conclude that while mitigation of (D)DoS attacks is possible at each stage, the area of focus of this research is to mitigate (D)DoS attacks at the (V) Ethernet switch. All the other stages are out of the scope of this research.

2.3 State-of-the-art on Ethernet switch

In the previous section, we examined the various attack surfaces that hackers can exploit to carry out (D)DoS attacks on IVN. We identified the TCU as a particularly important component to protect, as it acts as a gateway to the IVN and has a greater Internet reach. To defend against (D)DoS attacks on the IVN, it is necessary to examine strategies at the backbone of the network, where TCUs and other electronic control units (ECUs) are interconnected. An Ethernet switch is a key component of the IVN's backbone, and in this section, we will focus on its capabilities for mitigating (D)DoS attacks and answering the RQ2 (**What are the Ethernet switch's capabilities for mitigating (D)DoS attacks?**).

In this section, we will first discuss the operations of Ethernet switches in Sub-section 2.3.1. Next, we will discuss the general (D)DoS detection and prevention system in Sub-section 2.3.3 and in Sub-section 2.3.2, we will identify Ethernet switch capabilities to detect and mitigate (D)DoS attack. In Sub-section 2.3.4, we will conduct a review of the literature on methods used to detect and mitigate (D)DoS attacks in Ethernet switches. Finally, we will conclude the section with some final thoughts in Sub-section 2.3.5.

2.3.1 Ethernet Switch

The primary purpose of the Ethernet switch in an IVN is to relay Ethernet frames between the ECUs connected to a local network. From the OSI stack, the Ethernet switch lies on the

bottom two layers (i.e., the physical and data link layers). Each port can be individually configured at the physical layer for different operations. For example, automotive Ethernet 1000BASE-T1 twisted pair copper cables with a maximum range of 40 meters and 1 Gbps local data distribution can offer communication at the physical level. Automotive Ethernet is a networking technology that enables the connection of various ECUs within vehicles. It uses the Ethernet standard, which is commonly used in computers and other networking devices and relies on twisted pair cables or fibre optic cables to transmit data. Ethernet provides addressing methods using the Media Access Control (MAC) protocol to make sure the frames are received and transmitted to the correct ingress and egress ports. Figure 2.4 shows an example of an Ethernet switch's basic layout. The diagram illustrates the path that an Ethernet frame takes from the ingress port to the egress port, passing through the forwarding engine of the switch.

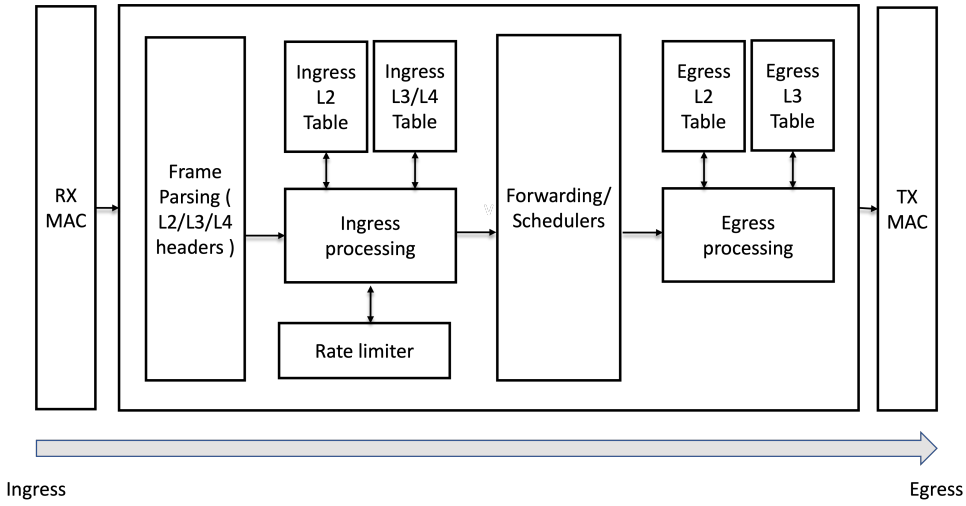


Figure 2.4: Simplified overview of Ethernet switch depicting forwarding process of Ethernet frame from an ingress port to an egress port

An Ethernet switch consists of various functions, including frame parsing, ingress processing, forwarding logic, and egress processing. The frame parsing stage extracts the header fields of the incoming frame, while the ingress and egress processing stages utilize various tables, such as L2-L4 tables, VLAN tables, and rate limiters, to determine how to handle the frame. The forwarding engine then applies the switch's forwarding logic to determine the corresponding egress port. The essential part of the forwarding engine is the L2 table, which includes MAC addresses attached to the corresponding port. The destination addresses are searched in the MAC table during the frame-forwarding process. The frame is forwarded to the egress port in case of a positive match. In case of no match is found, the frame is flooded to all ports, excluding the ingress port. The three primary functions of the Ethernet switch are described as follows:-

- MAC addresses learning and Source port enforcement - An Ethernet switch's primary function is to learn MAC addresses from incoming traffic. The source and destination

MAC addresses are included in the Ethernet frame headers. If the MAC addresses are not found in the MAC table, the MAC address is enforced to the table along with the appropriate ingress port.

- Forwarding tables - When a positive match in the forwarding table is found, the Ethernet switch forwards traffic to the corresponding egress port. In the case of a negative match, the frame is flooded across all ports to reach the destination address.
- Virtual Local Area Network (VLAN) - Separating traffic based on the groups or domains is an integral part of the network. Traffic separation can be performed in two ways; one is by adding a physical switch and creating another network for separate traffic. And another is to use the VLAN. VLAN is used to connect network devices regardless of their physical location logically.

With the Ethernet switch's basic functionalities, a frame can be forwarded over the network to the correct destination node. These functionalities only require processing the incoming frames by inspecting the L2 headers, such as source and destination MAC address, VLAN tags and Ether type. While this basic functionality has been sufficient for automotive Ethernet-based applications, security requirements necessitate an advanced level of packet inspection. Modern Ethernet switches provide the capability to limit or block traffic by utilizing ACLs and rate-limiters to prevent malicious traffic on the network. The next section describes how these techniques can be utilized to detect and prevent (D)DoS attacks.

2.3.2 Ethernet Switch Capabilities for Mitigating (D)DoS Attacks

ACLs and rate limiters are commonly used methods to mitigate (D)DoS attacks. The choice between the two may depend on the specific requirements and characteristics of the network and the type of (D)DoS attack being targeted [42].

ACLs are used to control access to a network by specifying which devices or traffic is allowed to pass through a network device, such as a gateway or switch. They can be used to block traffic from specific sources or destinations or to allow only certain types of traffic to pass through. ACLs can be effective at blocking (D)DoS attacks that involve traffic from a specific source or the type of (D)DoS traffic is known.

Rate limiters, on the other hand, are used to control the rate at which traffic is allowed to pass through a network device. They can be used to limit the amount of traffic that is allowed to pass through the device or to limit the rate at which traffic is allowed to pass through. While rate limiters can be effective at mitigating certain types of (D)DoS attacks, they have some potential disadvantages. These include the risk of false positives, where legitimate traffic is mistakenly identified as an attack and blocked or rate-limited. Rate limiters may also impact the performance of a network by limiting the amount of traffic

that is allowed to pass through, potentially leading to slower performance and reduced capacity [43].

Overall, both ACLs and rate limiters can be effective at mitigating (D)DoS attacks. However, the choice between the two may depend on the specific requirements and characteristics of the network, as well as the type of (D)DoS attack being targeted. For this research, which aims to mitigate known (D)DoS attacks with minimal false positive identification of legitimate traffic and minimal performance impact, the use of ACLs is relevant. While rate limiters can also be used to mitigate (D)DoS attacks, they are not included in this study and fall outside the scope of the research.

Access Control List (ACL)

An ACL uses a packet filtering mechanism to filter out packets based on the rules. The rule defines the conditions for the packet to be matched or not. These rules include fields to be matched, such as the packet's source address, a destination address, source port, and destination port. The purpose of the ACL is to accurately identify packets and prevent malicious traffic from occupying network bandwidth. In principle, the rules in the routing table are used for packet forwarding, whereas the rules in the ACL are used for packet classification. High-end switches use a memory type called Content Addressable Memory (CAM), where ACLs are stored. CAM is a type of memory where the user defines content or value and receives the message address. This is opposite to the functionality of the DRAM or SRAM, where the memory address is the input, and the content or value is the output.

An example of an ACL is shown in Listing 2.1. Source and destination IP address, source and destination port numbers, and protocol type are the five common fields seen in ACL entries. Packets are categorized by the relevant rule, and only when all fields match is a necessary action (permit/deny) taken [10]. For instance, in Listing 2.1, access- rule 3 allows TCP packets with source IP addresses from 10.1.1.1 to 10.1.1.255.

```
rule 1 deny tcp source 10.10.2.0 0.0.0.255 tcp-flag ack
rule 2 permit tcp source 10.10.2.0 0.0.0.255 tcp-flag rst
rule 3 permit tcp source 10.1.1.2 0.0.0.255 time-range time1
rule 4 deny tcp destination 192.168.2.2 0 fragment
rule 5 deny ip
```

Listing 2.1: Example ACL rule

2.3.3 (D)DoS detection and prevention

Some (D)DoS attack defence mechanisms are based on intrusion detection and prevention system (IDPS) and firewalls. IDPS and firewalls are both security solutions, but they

have different functions. An IDPS is designed to detect and prevent malicious activity on network traffic by analyzing patterns and behaviours that indicate an attack is taking place. Once an attack is detected, the IDPS can take action to prevent it from continuing, such as blocking the source of the attack. A firewall filters incoming and outgoing network traffic according to pre-defined rules. It can block undesirable traffic, such as from known malicious IP addresses, thereby helping to prevent (D)DoS attacks. IDPS are often utilized in conjunction with firewalls, serving as an additional layer of security.

This section will provide the advantages and disadvantages of different types of IDPS systems, such as signature-based detection and anomaly-based detection, and different types of firewalls, such as stateful and stateless firewalls.

- **Signature-based detection**: Rule-based detection, pattern detection, and knowledge-based detection are other names for a signature-based detection approach. This technique analyzes incoming traffic with a signature database to identify various attacks. The signature database comprises predefined signatures that contain attack patterns. It is only able to identify known attacks. The primary benefit of signature-based detection is that it uses less processing power and generates fewer false positive alerts. This method's drawback is that it cannot detect unidentified attacks and necessitates frequent updating of the signature database [44].

- **Anomaly-based detection**: Behavior-based detection is another name for anomaly-based detection. This technique identifies abnormal behaviour by identifying patterns or deviations from legitimate traffic. This method involves building a profile of legitimate behaviour by analyzing historical data, and then using this profile to identify patterns or events that deviate from the legitimate traffic. The key benefit of anomaly-based detection over signature-based detection is that it can identify unidentified attacks. This method's high rate of false positive alarms is a drawback [45].

- **Firewall (Stateful)**: A stateful firewall blocks traffic based on the flow states of the packets. For instance, a TCP protocol requires a three-way handshake in order to establish a connection between two devices. The firewall keeps track of each TCP handshake and blocks the packets belonging to the exciting flow in case of (D)DoS attacks. During fragmentation attacks, the firewall blocks the entire packet in which a malicious fragmented packet is found. The advantage of a stateful firewall is that it protects the device from malicious open connection states and malformed packets. However, compared to the stateless firewall, stateful firewalls are slower because it requires to have knowledge of the flow.

- **Firewall (Stateless)**: Stateless firewalls block traffic based on the header field of the packets. It does not particularly require knowing the flow. The access control

rule is applied to each packet arriving at the Ethernet port. Stateless firewalls are faster and utilize fewer resources compared to stateful firewalls.

Overall, both intrusion detection and prevention systems (IDPS) and firewalls are effective in mitigating (D)DoS attacks, but they each have their own strengths and limitations. By combining an anomaly-based IDPS and a stateless firewall, the benefits of both methods can improve their overall defence against (D)DoS attacks. Anomaly-based detection allows for the identification of previously unknown attacks, while the stateless firewall provides a fast and resource-efficient solution to block detected unwanted traffic.

2.3.4 Literature review

There are various security technologies used to secure automotive Ethernet. Lue et al. [46] segregate this technology into three criteria: isolation and filtration, detection and defence, and authentication and encryption. One of the basic security features of the network is the firewall. A firewall at the hardware level uses TCAM to filter traffic. Medhi et al. [47] show how TCAM filters traffic based on IP packets. Wang et al. [48] propose a TCAM search framework to identify TCP attacks based on the spoofing of IP addresses. On the other hand, firewalls can also be implemented in software. Many researchers [49] [50] [51] provide a mechanism to defend against the attack using software-defined networking (SDN), but software-based ACL requires all traffic to be directed to the host processor. This results in utilizing the processor's resources and is slower compared to hardware-based ACL [10].

In addition to firewalls, researchers have also investigated the IDPS system firewall because a firewall only permits traffic based on rules, not traffic patterns. Lin et al. [52] proposed a design to use Network Function Virtualization (NFV) to lessen the traffic overhead to the controller using an IDPS system. Chung et al. [53] extend the IDPS system to measure the time delay and the throughput of the packets transferred along the network. Yan et al. [54] propose a (D)DoS detection system with a design approach based on fault tolerance. Apart from filtering and detection techniques, researchers have also investigated anomaly-based (D)DoS detection methods. (D)DoS attacks cause a sudden surge in network traffic and can be detected by monitoring the randomness in the network traffic. A Shannon entropy measures the uncertainty associated with a random variable in information theory. Aladaileh et al. [55] uses Shannon entropy to detect network traffic abnormality. They calculate the joint entropy of header features like source IP, destination IP and protocol value and conclude that the method effectively lowers the false positive rate, reduces complexity and increases detection precision.

2.3.5 Concluding remarks

This section aimed to provide a functional overview of Ethernet switches and their capabilities to mitigate (D)DoS attacks. To explain this clearly, we discussed the Ethernet switch's forwarding pipeline and identified that the Ethernet switch provides packet inspection capabilities for inspecting network traffic. We identify signature-based and anomaly-based detection as types of packet inspection methodology. We surveyed the Ethernet switch's ACL capabilities to block the (D)DoS packet based on header inspection. Finally, we conducted a literature review on attack defence mechanisms.

2.4 Conclusion

This chapter aims to address two research questions: RQ1 **(What are the potential entry points for a hacker to gain access to an in-vehicle network?)** and RQ2 **(What are the Ethernet switch's capabilities for mitigating (D)DoS attacks?)**. In order to answer the first question, we conducted a survey of the various entry points of an in-vehicle network and identified the entities that are vulnerable to network attacks. We classified hackers based on the range of access they have (i.e., long-range, short-range) and the medium (e.g., OBD II port) that they use to access the network. To answer the second question, we conducted a background study on (D)DoS attacks and identified the different stages at which they can be detected and prevented. We also analyzed strategies for detecting and preventing such attacks and examined the use of ACLs by Ethernet switches to mitigate (D)DoS attacks.

Chapter 3

Design of Ethernet switch lookup behaviour

The previous chapter analyzed the potential entry points for hackers to gain access to an IVN and how these entry points can be used to launch (D)DoS attacks. We identified the different levels at which attack vectors can be detected and prevented. We focused on analyzing (D)DoS mitigation strategies at the Ethernet switch level. To provide the (D)DoS detection and prevention framework, it is first relevant to model the Ethernet switch pipeline. So the objective of this chapter is to answer RQ3 (**How to replicate the Ethernet switch's lookup behaviour in the simulation environment?**). To do this, we use the MATLAB tool to create a simulation environment that can take input in two formats: a C array and a CSV file.

We used the Wireshark tool to convert a PCAP file (a capture of network traffic) into a C array, which represents an Ethernet frame as an array of bytes. This will allow us to analyze the behaviour of an Ethernet switch's lookup process by demonstrating the design of the switch's lookup behaviour. We also used the Wireshark tool to convert the PCAP file into a CSV file, which we will use to evaluate our method for detecting and preventing (D)DoS attacks using publicly available datasets. The publicly available datasets will allow us to test and understand the effectiveness of our method.

Figure 3.1 illustrates the design of the Ethernet switch's existing lookup capabilities, which has been adapted from [56]. The design consists of three main blocks: the parser, key generation, and ACL lookup table. The parser block receives an Ethernet frame as input, extracts relevant information such as source and destination MAC addresses, and outputs this information. The key generation block processes this information to determine the key used to search the ACL lookup table. The ACL lookup table stores rules that define actions to be taken for certain types of traffic. When a key is received, it searches for a matching rule and outputs the specified action, such as forwarding or dropping the frame. Each block's input, process and output are illustrated in this chapter. It should be noted

that this design only implements the lookup behaviour of the existing Ethernet switch, and not all the Ethernet switch functionalities have been implemented. In Chapter 4, we will use this design as a basis for proposing a (D)DoS detection and prevention framework for an Ethernet switch.

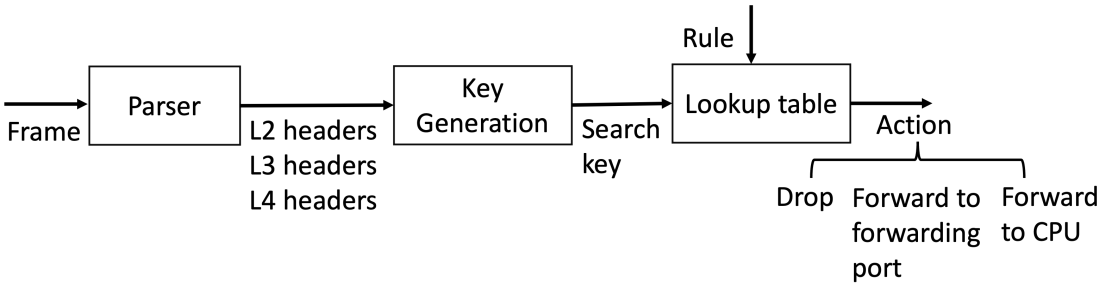


Figure 3.1: Simplified overview of existing switch Ethernet switch's lookup capabilities [56]

3.1 Parser

The parser is the first component of the forwarding process of the Ethernet switch. It is responsible for extracting the header field of an Ethernet frame. Figure 3.2 represents the input and output of the parsing block.

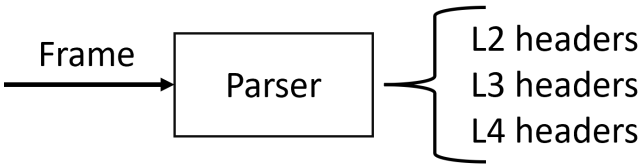


Figure 3.2: Overview of the parsing block

3.1.1 Input

The input of the parser block is an Ethernet frame. The Ethernet header length depends on whether the frame consists of a single VLAN tag, a double VLAN tag or no tag. A standard Ethernet frame is depicted in Figure 3.3. Before the payload, all the fields in the Ethernet frame are considered header fields. The first field of the Ethernet frame is the preamble. This field indicates the start of the Ethernet frame and is stripped off at the OSI layer 1. The next field is the first index of the frame, which is the destination MAC address (L2DA) followed by the source MAC address (L2SA). The MAC addresses must be unique in the same network to avoid communication problems between source and destination devices. The last field of the layer 2 header is the ethertype (ETH) value. The ETH value has a length of 2 bytes and represents the type of protocol which is used to send the payload. The typical example of the ETH value is 0x0800, which represents the

IPv4 packet. Apart from the standard header fields, there is one exception of the VLAN tag (802.1Q tag). In this case, the header fields are extended with 4 bytes or 8 bytes for a single or double VLAN tag, respectively. The VLAN tags are traditionally used to connect devices virtually and irrespective of their physical location. As shown in Figure 3.3, an extra VLAN tag field of four bytes is added to the frame.

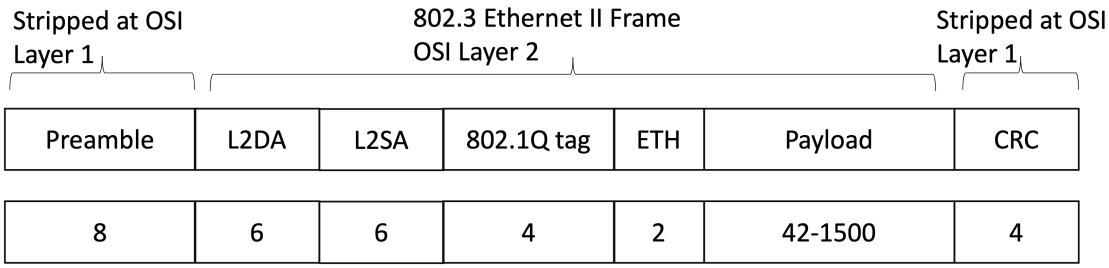


Figure 3.3: Example of 802.3 Ethernet II frame with single VLAN tag

The parser block receives an array of bytes as input, so it is unaware of the header fields or payload fields within the Ethernet frame. To demonstrate the process, we captured a sample network frame using the Wireshark tool and converted it into a C array using the same tool. An example of this captured frame in a C array is shown in Listing 3.1. The array has a total length of 66 bytes. We then convert this captured packet into a MATLAB array, which serves as the input for the parser block.

```
// Frame (66 bytes)
static const unsigned char pkt1[66] = {
0x02, 0x05, 0x85, 0x7f, 0xeb, 0x80, 0x02, 0x00,
0x00, 0x00, 0x00, 0x00, 0x08, 0x00, 0x45, 0x00,
0x00, 0x34, 0x0d, 0xb4, 0x40, 0x00, 0x78, 0x06,
0xfb, 0x55, 0x86, 0x1b, 0xe0, 0x5a, 0x0a, 0x8b,
0x88, 0xb9, 0x00, 0x50, 0xc8, 0x24, 0xee, 0x46,
0x8e, 0xdb, 0xe9, 0x40, 0x54, 0x39, 0x80, 0x12,
0x20, 0x00, 0xd2, 0x34, 0x00, 0x00, 0x02, 0x04,
0x05, 0xb4, 0x01, 0x03, 0x03, 0x08, 0x01, 0x01,
0x04, 0x02                 };
```

Listing 3.1: Example raw Ethernet frame from Wireshark capture

3.1.2 Process

Parsing is the process of extracting the header of an Ethernet frame. It works on the principle of first identifying each header based on the preceding header. For example, to parse the layer 3 header, the parser should know the Ethertype field value from the layer 2 header. Therefore, the entire process of parsing is sequential. Figure 3.4 depicts a simplified process of the parser. The arrow represents the index of the parser, and the

state shows the header and length fields known by the parser at that state. It shows the step-by-step process of parsing until layer 4 headers.

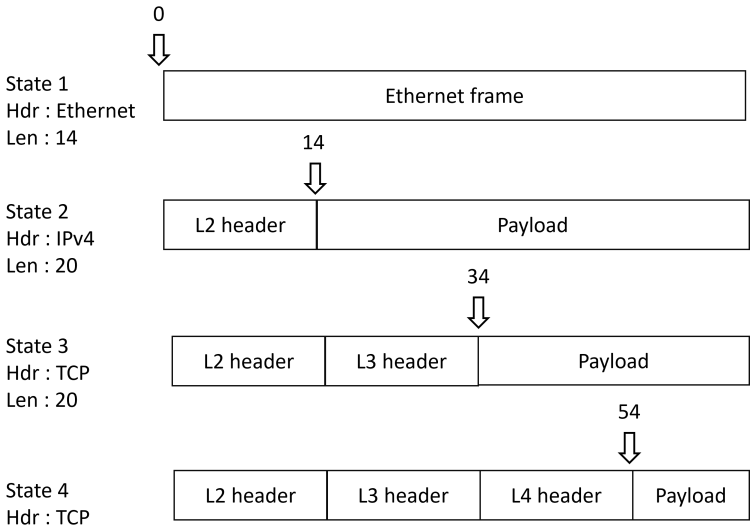


Figure 3.4: Simplified overview of parsing mechanism

The first level of parsing begins at the start of the frame. The initial header type is known by the parser. As seen in Algorithm 1, the first 6 bytes are extracted as L2DA, followed by the L2SA of 6 bytes. The next field depends on whether or not the VLAN tag is present in the Ethernet frame. For example, if it is a single tagged frame, the parser will parse the 17th byte for the Ethertype value, and if there is no tag, the parser will parse the 13th byte for the Ethertype value. Similarly, layer 3 and layer 4 parsing to extract the header field is performed.

Algorithm 1 L2 parsing

```
1: L2DA = frame(1:6);
2: L2SA = frame(7:12);
3: if No Tag then
4:     Ether Type = frame(13:14);
5: else if Single || Outer tag then
6:     Tag= frame(13:16);
7:     Ether Type = frame(17:18);
8: else if Double tag then
9:     Outer tag= frame(13:16);
10:    Inner tag= frame(17:20);
11:    Ether Type = frame(21:22);
12: end if
```

3.1.3 Output

The output of the parser block is a set of headers for each incoming Ethernet frame. To demonstrate the process, we have logged these headers in a .txt file, as shown in Listing

3.2. This listing shows the parsed output of the same Ethernet frame that was shown in Listing 3.1 as the input to the parser block. The key generation block uses these headers to generate the search key that will be used to search the ACL lookup table.

```
L2DA = [02 05 85 7F EB 80];  % 02:05:85:7F:EB:80

L2SA = [02 00 00 00 00 00];  % 02:00:00:00:00:00

ETH =  [08 00]; % 0x0800

IPv4_hdr = [69 0 0 52 13 180 64 0 120 6 251 85 134 27 224 90 10 139
    136 185]; % First 20 bytes of Layer 2 payload

TCP_hdr = [0 80 200 36 238 70 142 219 233 64 84 57 128 18 32 0 210
   52 0 0]; % First 20 bytes of Layer 3 payload
```

Listing 3.2: Output of the parsing block

3.2 Key Generation

The Key generation block uses information from the parser block to generate a search key. The search key is composed of the frame's first 16/32/64/128 bytes, depending on the header of interest. To mitigate (D)DoS attacks based on IP headers, we need at least the first 64 bytes to extract the source IP, destination IP, and protocol value from the Ethernet frame. The key generation process involves putting all the relevant header fields in the tuples to create a search key. After generating the search key, the search key is used by the ACL lookup table to find the match. Figure 3.5 represents the overview of the Key generation block.

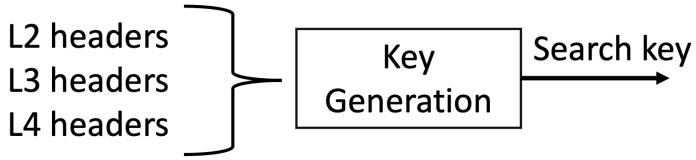


Figure 3.5: Overview of the Key generation

3.2.1 Input

As mentioned earlier, the input of the key generation block is the output from the parsing block. To extract the source IP address, destination IP address, source and destination port numbers, and protocol type for the ACL lookup table, we need to extract the first 64 bytes of the frame. However, from the parser, we do not get particular header fields like

the source IP of the frame. Therefore, the key generation block's first step is to extract the relevant headers. We recall the output of the parsing block as an input to the key generation block in Listing 3.3.

```
L2DA = [02 05 85 7F EB 80];  % 02:05:85:7F:EB:80

L2SA = [02 00 00 00 00 00];  % 02:00:00:00:00:00

ETH =  [08 00]; % 0x0800

IPv4_hdr = [69 0 0 52 13 180 64 0 120 6 251 85 134 27 224 90 10 139
    136 185]; % First 20 bytes of Layer 2 payload

TCP_hdr = [0 80 200 36 238 70 142 219 233 64 84 57 128 18 32 0 210
   52 0 0]; % First 20 bytes of Layer 3 payload
```

Listing 3.3: Input of the Key generation block

3.2.2 Process

The key used by the lookup table is generated using the key generation block. The composition of the key depends on the header of interest. To identify layer 4 (D)DoS attacks, we need at least the source IP address (L3SA), destination IP address (L3DA), source port (SP), destination port (DP) and protocol fields of the Ethernet packet.

As depicted in Algorithm 2, we first extract the relevant header fields from the parser output to create a key. For this example, the key is created based on the L3SA, L3DA, and protocol values from the IPv4 header. Source and destination port values from the TCP or UDP headers. The key is further used by the ACL lookup table to find the match with the rule.

Algorithm 2 Key generation process

```
L3SA = IPv4_hdr(13:16);
L3DA = IPv4_hdr(17:20);
Protocol = IPv4_hdr(10);
if Protocol == 06 then
    SP = TCP_hdr(1:2);
    DP = TCP_hdr(3:4);
else if Protocol == 17 then
    SP = UDP_hdr(1:2);
    DP = UDP_hdr(3:4);
end if
Output : Search key == < L3SA >< L3DA >< Protocol >< SP >< DP >
```

3.2.3 Output

The key generation block's output is the key used by the lookup table. The key is composed of 13 bytes. The output is a tuple with the header fields as follow:

Search key $== < L3SA >< L3DA >< Protocol >< SP >< DP >$

Listing 3.4 represents the key generated by the parsing block. The first 4 bytes of the key represent the source IP (10.10.10.2), the next 4 bytes represent the destination IP (10.10.10.185), the next 1 byte represents the protocol number (6), the next 2 bytes represent the source port (80) and the last 2 bytes represent the destination port (0502).

```
Search key {
    L3SA = [10 10 10 2];          % source IP 10.10.10.2
    L3DA = [10 10 10 185];        % destination IP 10.10.10.185
    Protocol = [06];              % TCP
    SP = [80];                    % source port
    DP = [502];  }                % destination port
```

Listing 3.4: Output of the Key generation block

3.3 Lookup table (LUT)

In principle, ACL is stored in the lookup table (LUT). The LUT table uses input from the key generation block, specifically a key generated using the header fields. In addition, LUT requires a rule that specifies action if the key matches the ACL. This rule is defined to prevent security threats. The output of the LUT is an action associated with the matched key. If the key matches the rule, the typical action is to deny or permit the frame to the destination address. Figure 3.6 represents the overview of the lookup table block.

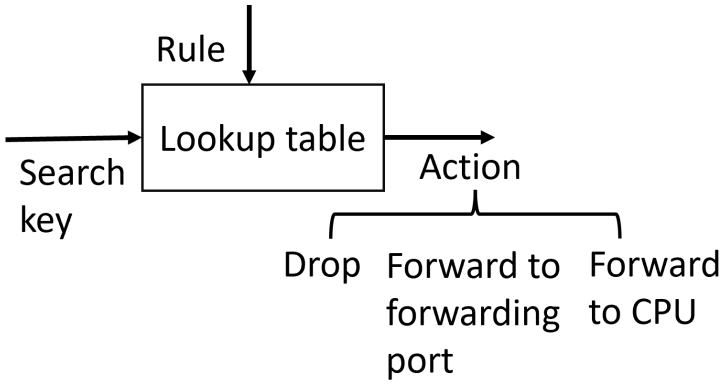


Figure 3.6: Overview of the Lookup table

3.3.1 Input

The lookup table has multiple inputs: the key generated using the key generation block and the ACL rule. We will assume a rule for this example and use the key from the key generation block. In the next chapter, we will describe the ACL rule generation process. We recall the output key from the key generation block and define an example rule in Listing 3.5 and 3.6, respectively.

```
Search key {
    L3SA = [10 10 10 2];          % source IP 10.10.10.2
    L3DA = [10 10 10 185];        % destination IP 10.10.10.185
    Protocol = [06];              % TCP
    SP = [80];                    % source port
    DP = [502];                   % destination port
}
```

Listing 3.5: Search key: Input of the ACL Lookup table

```
Rule {
    L3SA = [x x x x];       % source IP (0.0.0.0 to 255.255.255.255)
    L3DA = [10 10 10 185]; % destination IP 10.10.10.185
    Protocol = [06];        % TCP
    SP = [x];               % source port (0 to 65536)
    DP = [502];             % destination port
}
action = drop;
```

Listing 3.6: Example rule as an input of the Lookup table

3.3.2 Process

The ACL matching mechanism consists of a search key and a rule. In particular, if the search key matches the rule, the action is defined to allow or drop the packet. The key is first checked with the first rule defined in the ACL. If the key does not match the rule, the key is then checked with the next rule. The process is repeated until the key is matched with any of the rules defined. If no match is found, it flags the negative match. If the key is matched with any of the rules, the action associated with the match can be performed. It should be noted that the number of rules which could be defined in ACLs is limited and therefore requires an optimised way to generate rules. In the next chapter, we will discuss ACL rule generation in detail.

3.3.3 Output

The output of the ACL LUT is the action that is defined in the LUT. The typical action associated with a packet in LUT is to permit or deny the packet. The permit action allows the packet to be further processed in the switch. Listing 3.7 shows an example of the output as the action to drop the packet if the search key matches the ACL rule.

```
match {
    L3SA = [x x x x];       % source IP (0.0.0.0 to 255.255.255.255)
    L3DA = [10 10 10 185]; % destination IP 10.10.10.185
    Protocol = [06];        % TCP
    SP = [x];               % source port (0 to 65536)
    DP = [502];             % destination port
}
then drop;                  % action
```

Listing 3.7: Output of the ACL block

3.4 Conclusion

This section's aim was to answer RQ3 **(How to replicate Ethernet switch behaviour in the simulation environment ?)**. To demonstrate the lookup behaviour of a switch, we split the design into three blocks. In block 1, we mimic the parser's behaviour by first investigating the parsing mechanism and then implementing it. We perform the parsing logic to extract header fields from layer 2 to layer 4 headers. In block 2, we perform the key generation process by identifying relevant header fields generated by the parser. The final block of the design chapter is the ACL Lookup table; The block is responsible for checking if the search key matches the rule. The search key is generated by the Key generation block, and the rule is defined by the user. Combined with these three blocks, we could demonstrate the lookup behaviour of the Ethernet switch in the simulation environment. After mimicking the behaviour of the Ethernet switch, it is relevant to assign the rule for the ACL Lookup table. This brings us to the following research question, which is RQ4 (How to identify known (D)DoS attacks and define ACL rules for Ethernet switches?). In Chapter 4, we will analyse the traffic pattern of a (D)DoS attack and define the ACL rule.

Chapter 4

Entropy based (D)DoS defence scheme

The previous chapter demonstrated the lookup behaviour of the Ethernet switch process. We identified that the Ethernet switch ACL lookup table requires an ACL rule to block malicious packets. Recall from Sub-section 2.3.2 that one of the purposes of the ACL rule is to accurately identify packets and prevent malicious traffic from occupying network bandwidth. To define an ACL rule to mitigate a (D)DoS attack, it is relevant first to inspect packets to identify (D)DoS attack. So in this chapter, we first provide the algorithm to analyse traffic patterns to identify the (D)DoS attack.

Traffic analysis can be done using two methods - statistical or machine learning-based. Statistical solutions are more efficient as machine learning-based methods have a heavier computational load and require challenging data. Additionally, current statistical solutions have limitations such as only having rate-limiting functions for (D)DoS detection [57]. Since Ethernet switches have limited computational resources, it's relevant to have a statistical (D)DoS detection and mitigation mechanism that has a manageable computational complexity.

Entropy is a statistical measurement that measures the randomness in a dataset. During a DDoS attack, attackers send similar packets that make up a significant amount of the traffic, leading to a decrease in the randomness of packet properties. Thus, our (D)DoS detection algorithm uses Shannon entropy to identify the abnormality in the traffic pattern. After detecting the (D)DoS attack, the packets are collected to create a fingerprint in the form of a (D)DoS summary. Based on the summarized (D)DoS attack, we provide the algorithm to create an ACL rule. The Ethernet switch lookup table utilizes this ACL rule to mitigate the (D)DoS attack. With this, the chapter aims to answer RQ4 (**How to identify known (D)DoS attacks and define ACL rules for Ethernet switches?**).

Figure 4.1 depicts the simplified overview of traffic analysis to perform (D)DoS detection

and ACL rule generation. The methodology is divided into three parts: Collecting (D)DoS data sets, performing network analysis, and generating ACL rules based on the network characteristic described in Sections 4.1, 4.2 and 4.3, respectively. Based on the findings, we will propose the (D)DoS detection and prevention framework in Section 4.4.

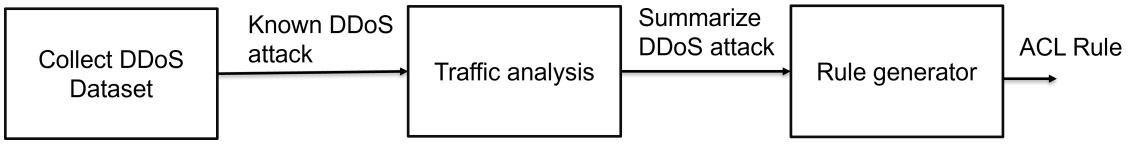


Figure 4.1: Simplified overview of traffic analysis for (D)DoS detection and dynamic generation of ACL rule for (D)DoS mitigation

4.1 (D)DoS dataset

The traffic analysis dataset is from publicly available datasets provided by ICS Cybersecurity 2018 [58] and CIC-DDoS 2019 [59]. The ICS Cybersecurity provides datasets for (D)DoS attacks on the SCADA system. It includes ICMP ping flood attacks and TCP SYN flood attacks. The CIC-DDoS 2019 is a dataset publicly published by the Canadian Institute for Cybersecurity. It contains 13 different types of (D)DoS attacks, out of which we use UDP flood and TCP-SYN flood attacks to perform traffic analysis. As mentioned in Section 2.2.1, UDP, TCP and ICMP are three internet protocols used in the automotive Ethernet, so for our analysis, only the attack vectors based on these three protocols are utilized.

The main difference between the ICS Cybersecurity 2018 and CIC-DDoS 2019 dataset is that the ICS Cybersecurity 2018 is comparatively smaller in scale (i.e., the maximum packet rate of TCP-SYN flood attack from is 200 PPS while the maximum packet rate of TCP-SYN flood attack from CIC-DDoS 2019 is 24941 PPS). All the (D)DoS datasets collected were in the PCAP file.

Before the traffic analysis process is presented, Tables 4.1 and 4.2 show the protocol distribution of TCP-SYN flood from CIC-DDoS 2019 and ICS Cybersecurity 2018, respectively.

Item	Packets	Percentage
TCP	2606280	99.4 %
ICMP	16442	0.6 %
UDP	0	0%

Table 4.1: Protocol distribution of TCP-SYN flood from CIC-DDoS 2019

Item	Packets	Percentage
TCP	79047	97.3 %
ICMP	0	0 %
UDP	66	0.1%

Table 4.2: Protocol distribution of TCP-SYN flood from ICS Cybersecurity 2018

4.2 Traffic analysis

The role of traffic analysis is to detect the (D)DoS attack using Shannon entropy. The ACL rule generation block will then utilize the detected (D)DoS packets to create an ACL rule. One method for detecting (D)DoS attacks is threshold-based detection, which involves setting a predetermined threshold value to distinguish malicious traffic from legitimate traffic. In this approach, we propose using entropy-based analysis to detect (D)DoS attacks by examining the entropy values of various packet attributes, such as IP address and protocol. Shannon entropy, a measure of uncertainty in information theory, has been shown to be effective in detecting abnormal traffic patterns based on the probability distribution of these attributes [60]. Our methodology involves analyzing both legitimate and malicious traffic to understand the characteristics of Shannon entropy in each case.

4.2.1 Methodology

One of the applications of Shannon entropy is to perform (D)DoS detection by determining the probability distribution of attributes in the headers of network packets. These attributes may include the source IP address, the destination IP, the protocol, and other attributes that identify the packet's characteristics. For instance, the detection might record 1000 packets in a one-second time window and the frequency of each unique IP address among these 1000 packets. We could determine the randomness of these packets by performing entropy calculations on this distribution [61].

After researching the behaviour of (D)DoS attacks from the collected datasets, the assumption is made that there will be a significant amount of packets or incomplete connection requests when the attack starts. For instance, during legitimate traffic, the attributes of packets are typically dispersed randomly. So the entropy value of legitimate traffic is high. Whereas, during (D)DoS attacks, the randomness in some attributes tends to remain the same. For example, during a TCP-SYN flood attack, the TCP SYN flag will remain the same for the attack period. The decrease in the randomness of the attribute will decrease the entropy value. The change in entropy behaviour is used as data points to detect (D)DoS attacks.

We determine the entropy by inspecting the randomness in attributes of packets in one second time window. The entropy can range from 0 to log M, where M represents the number of packets evaluated. Entropy is zero when every distribution value is the same, whereas entropy is maximum when all the distribution values are different. The entropy of a random variable X is defined as H(X), where X denotes the values form x_1, x_2, x_3,...,x_M, and the probability of occurrence of X corresponds to P_1, P_2, P_3,...,P_M. Equation 4.1 provides the Shannon entropy.

$$H(X) = -\sum_{i=1}^{M} P(x_i)\, log_2\, P(x_i) \tag{4.1}$$

Where H is the entropy, M is the total number of packets being evaluated, and P_i is the probability of each unique attribute in a certain time window.

To calculate the entropy value of two attributes together, the joint entropy of the attributes X and Y is defined as in Equation 4.2.

$$H(X,Y) = -\sum_{i=1}^{M}\sum_{k=1}^{N} P(x_i y_k)\, log_2\, P(x_i y_k) \tag{4.2}$$

Where X and Y represent two attributes of a packet, P($x_i y_k$) is the joint probability of attributes X and Y when X = x_i and Y = y_k, M and N is the total number of packets in a certain time window.

To directly compare the entropy value of a single attribute and the joint entropy of multiple attributes, we determine the normalized entropy by limiting the entropy value between [0,1]. Equation 4.3 and 4.4 represent the expressions for normalized entropy ($\bar{H}$) and normalized joint entropy ($\bar{JH}$), respectively.

$$\bar{H} = \frac{H(X)}{log_2 N} \tag{4.3}$$

$$\bar{JH} = \frac{H(X,Y)}{log_2 MN} \tag{4.4}$$

In this study, all entropy values are normalized. Therefore, when we use individual entropy or joint entropy terms, it means the value has been normalized.

The joint entropy of the attribute pair is initially determined by evaluating legitimate traffic, which we refer to as JH_n. The joint entropy of the current traffic is calculated at run-time and referred to as JH_c. To compare the values of JH_n and JH_c, we use Equation 4.5 to identify any decrease in the joint entropy value.

$$\Delta JH = JH_n - JH_c \tag{4.5}$$

A predefined threshold value is used to determine if the change in entropy, represented by ΔJH, exceeds an acceptable level. If ΔJH exceeds the threshold, it indicates a potential (D)DoS attack. The threshold value can be adjusted based on the system's security requirements. In Section 4.4, we will present an algorithm for entropy-based (D)DoS detection using this approach.

4.2.2 Experimental results and discussion

In this experiment, we aim to test the TCP SYN flood attack by calculating the individual entropy of various attributes, including the source IP, destination IP, protocol, source port, destination port, TCP flag, and packet length, as well as the joint entropy of selected attributes. It's important to note that while the analysis of individual entropy is useful for understanding the behaviour of each attribute individually, our methodology relies on the joint entropy of the selected attributes for detecting potential (D)DoS attacks.

To conduct the experiment, we collected 1800 seconds of traffic from the dataset provided in [58]. Out of 1800 seconds of collected traffic, the (D)DoS attack starts from the 300th second to the 1200th second. The selected time window size is 1 second, and the packets are divided into each time window. Thereafter, we calculate the entropy value of each attribute using Equation 4.3 for all time windows.

We normalized the entropy values and plotted the results on a line graph, with the vertical axis representing the value of entropy and the horizontal axis representing the 1-second time window. The findings for both the individual and joint entropy are presented below:

Individual entropy

- **Source IP entropy**: Hackers usually randomly allocate the source IP address to avoid getting backtracked. As a result, there may be instances where the deviation in unique source IP addresses is random during the (D)DoS attack. In our dataset (from [58]), we observed up to 200 unique source IP addresses in one second time window of the (D)DoS attack, and therefore, as seen in Figure 4.2 the entropy value increased significantly from 0.18 to 0.97 during (D)DoS attack. However, this is only a reflection of our data set. The unique source IP deviation could be random in a real-world scenario. Calculating entropy value based on the random behaviour of the source IP could be inefficient. As a result, we exclude the source IP entropy for (D)DoS detection.

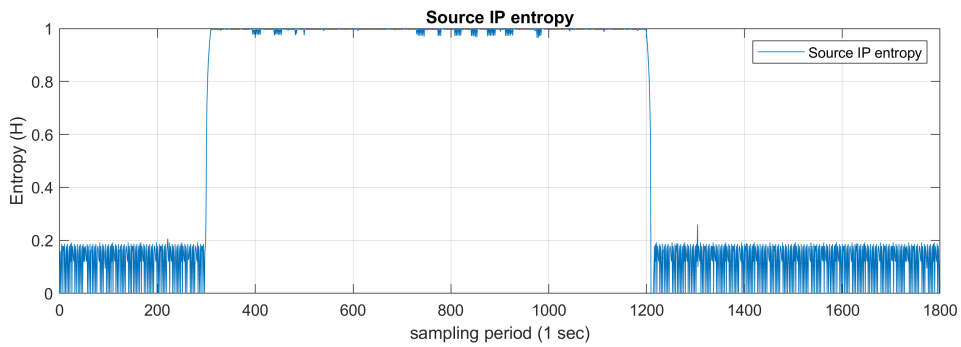


Figure 4.2: Source IP entropy

- **Destination IP entropy**: During the (D)DoS attack, the destination IP address is normally the victim's IP address. Therefore, the destination IP address remains the same for every spoofed source IP address. The re-occurrence of the same destination IP address reduces the entropy value. Figure 4.3 shows that the entropy value of the destination IP address is reduced from 0.72 to 0.06 during the (D)DoS attack. This is due to the fact that the randomness in unique destination IP remains the same during the attack period.

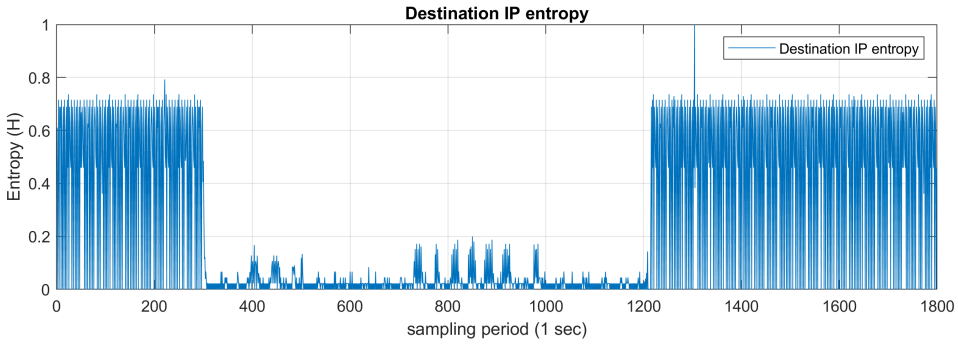


Figure 4.3: Destination IP entropy

- **Protocol entropy**: The protocol field of the TCP SYN flood attack is always representing TCP. During the attack, we receive a large number of packets containing the TCP protocol field. The entropy decreases as the randomness in the protocol field are less in one second time window. The behaviour of the entropy graph is depicted in Figure 4.4; we can notice the entropy drop, which is similar to the drop in entropy of the destination IP. However, in the case where the normal traffic has a large proposition of TCP packets, the randomness in protocol entropy may not be enough to identify the threshold.

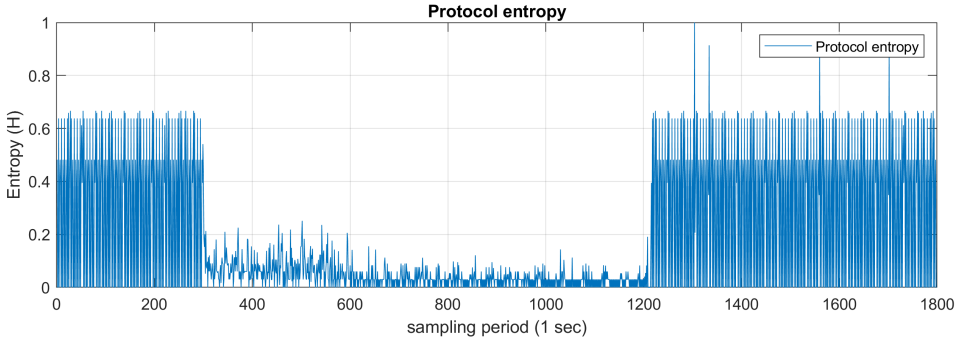


Figure 4.4: Protocol entropy

- **Source port entropy**: Similar to the source IP entropy, the source port can be spoofed by the hacker. Therefore the deviation in the source port depends on the randomness introduced by the hacker. Figure 4.5 shows that the entropy value increases from 0.17 to 0.98 during the (D)DoS attack. The increase in entropy is

due to the dataset containing distinct source ports during the (D)DoS attack. In a real-world scenario, the deviation in the source port may deviate depending on the hacker's implementation. Therefore there is a possibility that the entropy value is not significantly affected.

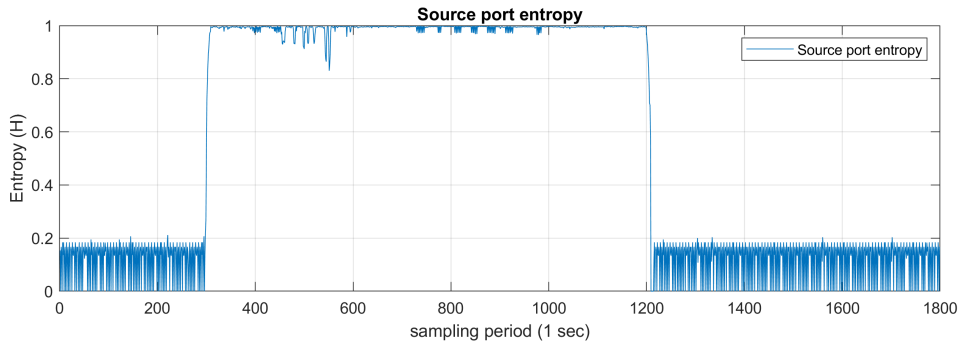


Figure 4.5: Source port entropy

- **Destination port entropy**: Destination port entropy depends on the unique destination port in a given window size. During the TCP SYN flood attack, the hacker usually tries to exploit a particular TCP port by sending a large number of TCP SYN packets. Therefore, the value of the TCP port remains the same during the (D)DoS attack and the entropy value of the destination port decreases during the attack. Figure 4.6 depicts the entropy calculation of the destination port.

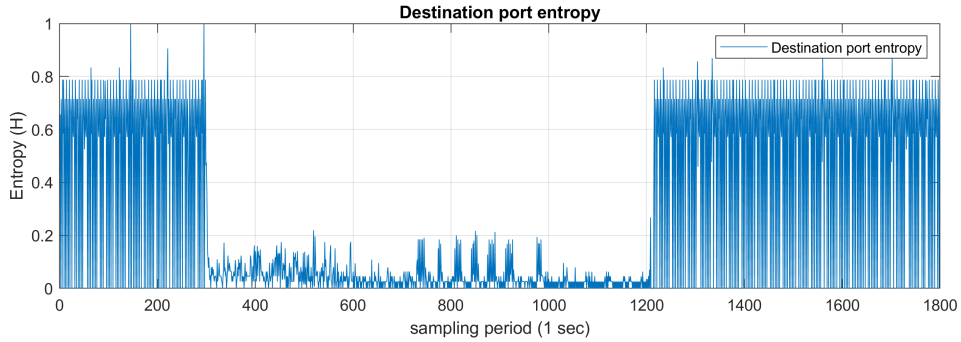


Figure 4.6: Destination port entropy

- **TCP flag entropy**: The hacker sends a large number of SYN packets to the destination device during a TCP SYN flood attack. The increase in packets with only TCP SYN packets reduces entropy. Figure 4.7 shows that during the (D)DoS attack, the entropy value of the TCP SYN flag decreases from 0.78 to 0.06. However, the destination device may process some of the TCP SYN packets and respond with the TCP SYN ACK packet. This response causes randomness in entropy values, which can be seen as small peaks between 300 and 1200 seconds.

- **Packet length entropy**: Flooding attacks often have identical packet sizes. For example, a normal-looking TCP-SYN packet has a packet length of 60 bytes. Hence,

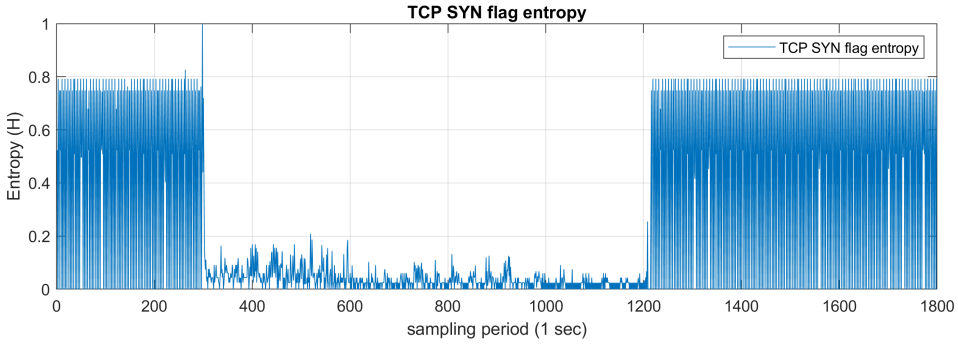


Figure 4.7: TCP flag entropy

the entropy in the length of the packet decreases during the TCP-SYN flood attack. From Figure 4.8, we can observe that the entropy of the packet length decreases during the (D)DoS attack period from 300 to 1200 seconds. This is because all the TCP-SYN packet captured from the dataset has the same packet length.

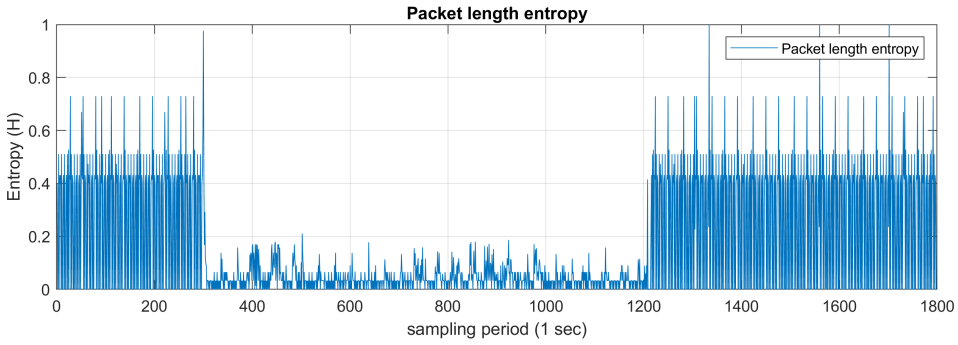


Figure 4.8: Packet length entropy

Joint entropy

After analyzing the entropy values of various attributes, we observed that during a (D)DoS attack, the entropy values for destination address, destination port, protocol, and flags decreased. To assess these values against a predefined threshold, we calculated the joint entropy using Equation 4.4. The joint entropy was calculated over a 1-second time window, and a moving average of 10 seconds was applied to eliminate short-term fluctuations.

In this experiment, we selected the pair of packet length and TCP flag as both attributes exhibited a decrease in individual entropy during the (D)DoS period. However, any attribute that demonstrates a decrease in individual entropy can be used to calculate joint entropy.

Recall that by calculating the difference between the entropy during the legitimate period (H_l) and the entropy during the (D)DoS period (H_c), we can identify the average decrease in entropy value (ΔH). Table 4.3 shows the ΔH for the individual entropy of the flag,

individual entropy of packet length, and joint entropy of flag and packet length. From the finding, we observed an average of JH_l (average joint entropy of legitimate traffic) of 0.854 and an average of JH_c (average joint entropy between 300th and 1200th seconds) of 0.073 during the (D)DoS period. Using Equation 4.5, we calculated the average decrease in entropy (ΔJH) value to be 0.781.

Attribute	Entropy (H_l)	Entropy (H_c)	ΔH
JE_Len_flag	0.854	0.073	0.781
E_Flag	0.820	0.083	0.727
E_Len	0.644	0.102	0.542

Table 4.3: Average decrease in entropy (ΔH) for individual and joint entropy

Figure 4.9 plots joint and individual entropy values for the packet length and TCP flag. It can be observed that during legitimate traffic, the joint entropy value is higher than the individual entropy values, while during (D)DoS periods, the decrease in joint entropy is larger than the decrease in individual entropy values. This suggests that the joint entropy method is more sensitive to detect (D)DoS attacks than individual entropy, and therefore our (D)DoS detection methodology relies on joint-entropy calculation.

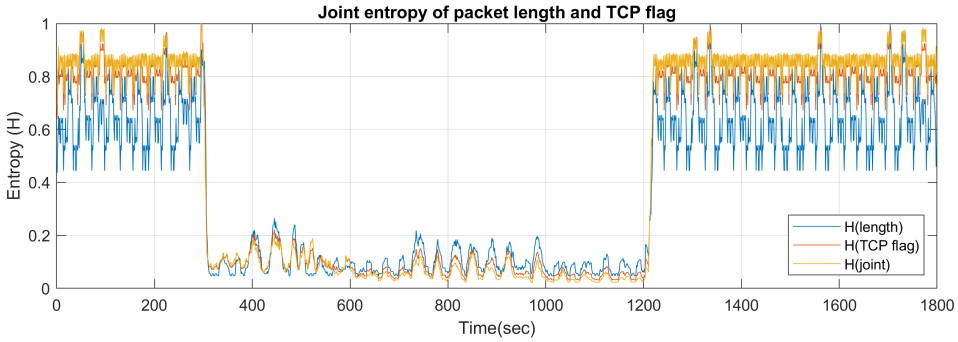


Figure 4.9: Joint entropy

In Chapter 5, we will use the same joint entropy results for different threshold value to evaluate the (D)DoS detection method.

4.2.3 Concluding remarks

In this study, we analyzed the entropy values of various attributes that may change during a (D)DoS attack. We specifically focused on the attributes of Source IP, Destination IP, Flags, Length, Protocol, Source Port, and Destination Port in the context of a TCP SYN flood attack. From our results, we found that the entropy values for the source IP and source port were not effective for detecting a (D)DoS attack, as the distribution of unique values for these attributes was generally random during the attack. On the other hand, we observed that the entropy values for the destination address, protocol, TCP flag, and

packet length decreased during the attack. To further evaluate our entropy-based (D)DoS detection methodology, we calculated the joint entropy of two selected attributes. The comparison of individual and joint entropy values showed that the joint entropy calculation was more sensitive for detecting a (D)DoS attack. Using the joint entropy approach, we were able to successfully detect the attack.

Following the detection of a (D)DoS attack, it is relevant to define an ACL rule to mitigate the attack. In the next section, we will present a methodology for defining such a rule.

4.3 Defining ACL Rule

In the previous section, we performed the network analysis and identified the header fields, which showed a change in entropy value during the (D)DoS attack. In this section, we use identified header field to create the ACL rule. The ACL rule is limited to the rule which can be defined using the header field. To generate an effective ACL rule, it is necessary that the ACL rule match as many header fields of (D)DoS packets as possible. For example, in the case of an ICMP ping flood attack, if an ACL rule is created only based on the protocol field, the rule will block both legitimate and malicious packets with the ICMP protocol. Whereas, if the ACL rule is created based on the destination IP, protocol, ICMP type, and ICMP code, The ACL rule will only drop the ICMP packets with the type echo request with the destination IP of the targeted device.

4.3.1 Methodology

ACL rule can be created using two approaches. The first approach is to define an ACL rule for each source IP address. The drawback of this approach is that the (D)DoS attacks may include many IP addresses, and memory-constrained devices are limited in the number of ACL rules that can be defined. The second approach is to define the ACL rule based on header fields which closely resemble the (D)DoS traffic. The drawback of this approach is that the ACL rule may discard legitimate traffic with the same header fields as (D)DoS traffic.

We take the second approach in our methodology because assigning an ACL rule for each source IP is not feasible in the resource-constrained Ethernet switch. The process for defining the ACL rule begins after our entropy-based (D)DoS detection methodology has detected an attack. Once the attack has been identified, the header fields of the (D)DoS packets are collected in a (D)DoS summary. The header fields included in our (D)DoS summary are based on the protocol field, as defined in Table 4.4.

- **Source IP**: The source IP address is the IP address of the exploited machine, which is used to generate (D)DoS attacks. During (D)DoS attacks, the source IP address

Protocol	Header fields
ICMP	{Destination IP} , {Protocol} , {ICMP type}, {ICMP code}
UDP	{Destination IP} , {Protocol} , { Source port }, {Destination port}
TCP	{Destination IP} , {Protocol} , { Source port }, {Destination port}, {TCP flags}

Table 4.4: Header fields used to create an ACL rule in the form of (D)DoS summary

is usually spoofed or originates from multiple sources. This was one of the main findings from our traffic analysis process. Meanwhile, the ACL rule is limited to creating a rule for one instance. This means if there are 1000 unique source IPs, we need to create 1000 ACL rules which define each of the unique source IPs. As a result, we exclude the source IP field from our (D)DoS summary.

- **Destination IP**: Destination IP denotes the victim's IP address. The (D)DoS summary always includes the destination IP header because the victim IP remains the same.
- **IP Protocol**: The attack vector is based on the particular protocol; therefore, for each attack vector, the protocol field remains the same. As a result, we always use the IP protocol field in our (D)DoS summary.
- **ICMP type**: The (D)DoS summary considers the ICMP type field when the attack vector is based on a specific type of ICMP packet (for example, ICMP echo packets).
- **ICMP code**: ICMP code is included when the attack vector also has ICMP code.
- **Destination port**: We consider this field when the attack vector is based on TCP or UDP protocol and has only one destination port.
- **Source ports**: We include the source port in the (D)DoS summary only when the attack vectors are generated from only one source port; if the attack vector is generated from multiple source ports, the (D)DoS summary excludes the source port field.
- **TCP flags**: The (D)DoS summary includes TCP flags only when the attack vector is based on the particular TCP flag (e.g., TCP SYN attack).
- **Packet length**: The packet length is included in the (D)DoS summary when the attack packets have the same packet length.

4.3.2 Experimental results and discussion

To create the ACL rule, we use the same dataset which we used to detect the (D)DoS attack in the previous section. After detecting the attack, the header fields are stored as the (D)DoS summary as defined in Table 4.4. After analysing the (D)DoS summary, it was

observed that the destination IP. Protocol value, destination port, TCP flag and packet length remained the same during the (D)DoS attack period. Based on this information, an example ACL rule is created as in Listing 4.1.

```
match {
    L3SA x.x.x.x;          % source IP (0.0.0.0 to 255.255.255.255)
    L3DA 172.27.224.250;   % destination IP
    Protocol 06;           % TCP
    SP x;                  % source port (0 to 65536)
    DP 502;                % destination port
    flag SYN;              % TCP flag
    packet length 175;     % length
}
then drop;                 % action
```

Listing 4.1: Example of ACL rule which is created to drop (D)DoS traffic

Since the ACL rule is created based on the (D)DoS packet's characteristics, it can always filter (D)DoS traffic. However, the ACL rule can affect legitimate traffic if they are not generated effectively. Therefore, we apply the ACL rule to legitimate traffic in this analysis.

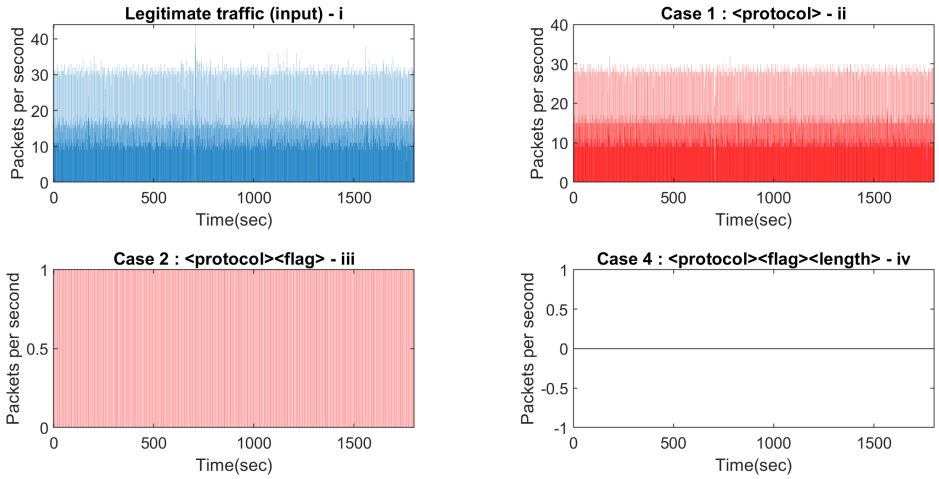


Figure 4.10: Legitimate packets as input (i) and three cases (ii-iv) of ACL rule when applied to the legitimate packets.

Figure 4.10 depicts the packet rate distribution of legitimate traffic and three scenarios of the ACL rule when applied to legitimate traffic. The description of each case is as follows:

- **Legitimate traffic (input)**: Figure 4.10(i) shows the distribution of packets in the legitimate traffic dataset. In total, 35430 legitimate packets were counted in the entire period of the legitimate traffic. This dataset is used to evaluate the ACL rule in the following 3 cases.

- **Case 1: ACL rule <protocol>**: Figure 4.10(ii) shows the packet count per second after the ACL rule based on the protocol field is applied to the legitimate traffic. It is observed that the ACL rule discarded a total of 32790 out of 35430 (92%) legitimate packets. This is because 92% legitimate traffic dataset is based on the TCP protocol, which is the same as the protocol value of a TCP-SYN flood attack. From the result, we conclude that defining an ACL rule based on just the protocol value should be avoided because it could affect legitimate traffic extensively.

- **Case 2: ACL rule <protocol><flag>**: Figure 4.10(iii) shows the packet count per second after the ACL rule based on the protocol and TCP flag field is applied on the legitimate traffic. It is observed that the ACL rule discarded a total of 621 out of 35430 (1.75%) legitimate packets. This is because 621 packets of legitimate traffic datasets are TCP-SYN packets, the same as the TCP flag value of the TCP-SYN flood attack. It should be noted that discarding the TCP-SYN packet will affect all the TCP packets associated with that particular TCP-SYN request packet. Dropping the TCP-SYN packets during the (D)DoS attack period is possible but not optimal.

- **Case 3: ACL rule <protocol><flag><length>**: Figure 4.10(iv) shows the packet count after the ACL rule based on the protocol, TCP flag and length field is applied on the legitimate traffic. It is observed that 0 out of 35430 (0%) legitimate packets were discarded by the ACL rule. This is because there is no TCP SYN flag packet in legitimate packets with a length size the same as defined in the ACL rule. We conclude from the result that the ACL rule is the most effective when it is created based on the fields that match as many fields of (D)DoS packets.

4.3.3 Concluding remarks

In this section, we examined the effectiveness of an ACL rule in mitigating a TCP-SYN flood attack. To do this, we first created an ACL rule based on certain characteristics of the attack traffic. We then evaluated the impact of this rule on legitimate traffic. Our results showed that an ACL rule that matches as many header fields as possible with the attack traffic is more effective at blocking the attack without also blocking legitimate traffic. Specifically, we found that an ACL rule based only on the protocol field had a significant impact on legitimate traffic, discarding 92% of legitimate packets. On the other hand, an ACL rule based on the protocol, flag, and length fields did not significantly impact legitimate traffic.

4.4 Proposed defence framework

In Chapter 3, we demonstrated the design of the Ethernet switch, which included the parsing, key generation and ACL lookup process. Chapter 4 introduced the traffic analysis using entropy calculation and ACL rule generation process. In this section, we will use all these components to provide a framework for the (D)DoS detection and prevention mechanism.

Figure 4.11 depicts an illustration of the proposed framework. The first step of the process is Ethernet frame parsing. Recall that the parsing block extracts the header fields of the input Ethernet frame. Based on the header fields, a search key is created to find a match with the ACL rule in the ACL lookup table. Following that, the ACL lookup table is responsible for distinguishing between known and unknown traffic. The known traffic is the packets with the IP addresses which are known to the ACL lookup table. In our case, the known traffic is the packets with the IP addresses of in-vehicle components that are preconfigured to allow communication with other in-vehicle components. Our methodology allows the known IP address to bypass our (D)DoS detection algorithm in order to prevent packet delay overhead which our algorithm can cause. Listing 4.2 (Rule 1) shows an example ACL rule which always allows packets from source IP 10.10.10.2 to destination IP 10.10.10.185. The unknown traffic is the traffic which is originated from an external device or a malicious device to communicate with in-vehicle components. Listing 4.2 (Rule 2) shows an example ACL rule for unknown traffic. This example ACL rule allows packets from any source IP to destination IP 10.10.10.185 and is created at the end of the ACL lookup table. As the ACL rule is located at the end of the ACL lookup table, the rule only matches after checking with all the ACL rules for known IPs. Every time this rule is matched, a counter is incremented. This allows the host CPU to keep track of the number of unknown packets. Packets arriving from external sources are usually not trusted and can cause (D)DoS attacks. So the main focus of our methodology is to perform (D)DoS detection and mitigation mechanism on the unknown traffic.

Starting from the unknown traffic, the counter value is incremented every time the unknown source IP is received. The host CPU monitors the counter value and resets it every second to monitor the rate at which the packets are received. The counter value is checked to determine whether it exceeds a predefined threshold (t_1). The packets are sent to the host CPU if the counter value exceeds the threshold (t_1). This is the first check to identify unusual traffic. Next, our entropy-based (D)DoS detection algorithm runs every time the packet is sent to the host CPU.

Algorithm 3 depicts the (D)DoS detection algorithm in which we identify the (D)DoS attack by calculating the entropy value. Here, we initialize the threshold values t_1 and t_2 for the counter and entropy threshold, respectively. Additionally, the attributes a_1 and a_2 are selected for joint-entropy calculation. JH_n is used as the indication for the entropy

value of legitimate traffic. The first step of the algorithm is to capture unusual traffic by comparing the counter value with the threshold (t_1). If the counter value exceeds the threshold, packets are sent to the host CPU for entropy calculation. The ACL counter does the first check in this algorithm. The next step is calculating the entropy value by using the joint-entropy formula on selected attributes a_1 and a_2. The threshold (t_2) is defined to represent the maximum decrease in the joint-entropy value that is allowed. The maximum decrease (ΔJH) is calculated using the Equation 4.5. Next, if ΔJH is greater than the threshold (t_2), the flag is created, suggesting (D)DoS detected, and the packets are considered as (D)DoS packets. The flag invokes our (D)DoS mitigation algorithm every time it detects the (D)DoS packets. Algorithm 4 presents the ACL-based (D)DoS mitigation algorithm. Here, after detecting the (D)DoS attack, we collect the header samples from the packets. Next, header fields are selected based on Table 4.4 and are summarised to create an ACL rule. The ACL rule is then configured to the ACL lookup table to drop the detected (D)DoS packets.

The current proposal does not address the potential for unknown traffic to be received at a rate that exceeds the processing capacity of the host CPU. To prevent this from occurring, it is relevant to use rate-limiters when sending packets to the host CPU. Additionally, after detecting and generating an ACL for (D)DoS packets, a combination of ACL lookup and rate-limiting can be implemented to mitigate (D)DoS attacks effectively. For example, during a UDP flood attack, an ACL rule can be created to detect UDP packets, and a rate limiter can be applied only to those packets. The proposal does not address the possibility of a (D)DoS attack resulting in an unusual increase in known traffic. However, this issue can be resolved by limiting the bandwidth available to the known source IP.

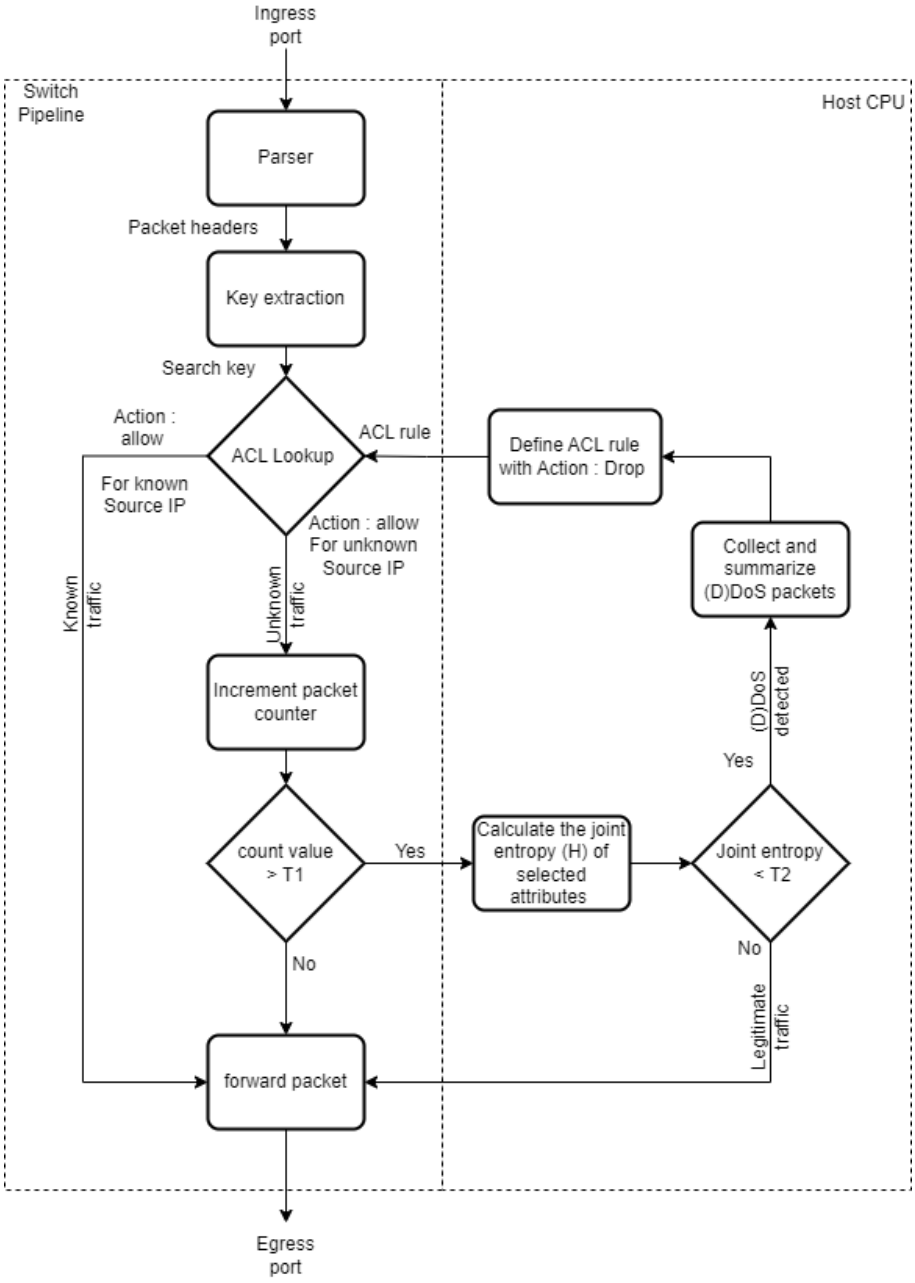


Figure 4.11: An illustration of (D)DoS detection and mitigation framework

```
match {  % Rule 1 - known packet
    L3SA = [10 10 10 2];          % source IP 10.10.10.2
    L3DA = [10 10 10 185];        % destination IP 10.10.10.185
}
then allow;                       % action
match { % Rule 2 - unknown packet
    L3SA = [x x x x];             % source IP (0.0.0.0 to 255.255.255.255)
    L3DA = [10 10 10 185]; % destination IP 10.10.10.185
}
then allow;                       % action
Counter ++;                       % increment counter
```

Listing 4.2: Example of preconfigured ACL rule to always allow known source IP and to count unknown source IP addresses

Algorithm 3 Entropy-based (D)DoS detection

Input: Collected (D)DoS datasets, selected attributes $a1$, $a2$, legitimate traffic entropy JH_n, Threshold t_1,t_2 for rate threshold and entropy threshold respectively
Monitor the counter value from the switch pipeline
if counter value $> t_1$ **then**
    Collect packet attributes
    Calculate the joint probability of selected attributes $a1$, $a2$
    Calculate current joint entropy(JH_c) using Equation 4.4
    Calculate decrease in joint entropy using $\Delta JH = JH_n - JH_c$
    if $\Delta JH > t_2$ **then**
        attack_flag = True
    else
        attack_flag = False
    end if
else
    Forward packet
end if
Output: A boolean value (attack_flag) representing the attack status

Algorithm 4 Defining ACL rule

Input: (D)DoS datasets, attack_flag from entropy based (D)DoS detection
if attack_flag = True **then**
    Collect header samples of (D)DoS packets
    Define ACL rule based on the header fields
    Assign action of ACL rule as drop
    Configure the ACL lookup table with the new ACL rule
else
    Assign action of ACL rule as allow
    Allow packets to the forwarding pipeline
end if
Output: ACL rule

4.5 Conclusion

This chapter aims to answer RQ4 (**How to identify known (D)DoS attacks and define ACL rules for Ethernet switches?)** . We approach this research question by first exploring two strategies for identifying known (D)DoS attacks. First, we calculate the individual entropy of various attributes in a TCP SYN flood attack, such as source and destination IP, protocol, source and destination port, TCP flag, and packet length. Our results show that entropy values generally decrease during (D)DoS attacks as the randomness of attribute values decreases. Next, we perform a joint-entropy calculation using packet length and TCP flag as attributes. The results indicate that joint-entropy values are more sensitive in detecting (D)DoS attacks compared to individual entropy. Based on these findings, we conclude that using joint entropy is an effective method for entropy-based detection.

Once the (D)DoS attack is detected, the packets are summarized in the form of (D)DoS summary. This summary is used to create an ACL rule. The ACL table is then updated with the ACL rule. In the second part of our investigation, we test this rule for legitimate traffic to evaluate its impact on network traffic. Our results show that the ACL rule is able to mitigate (D)DoS traffic, but it also inadvertently drops legitimate traffic with header fields similar to those of (D)DoS packets. We conclude that it is possible to mitigate (D)DoS attacks using an ACL rule, but to be effective, the rule must match as many header fields of (D)DoS packets as possible to minimize the number of legitimate packets discarded.

The chapter also provides an overview of a (D)DoS detection and prevention framework, which combines the Ethernet switch look-up design from the previous chapter with traffic analysis and ACL rule generation processes.

Chapter 5

Results and evaluation

In the previous chapter, we outlined the steps for analyzing network traffic and creating ACL rules to detect and prevent attacks. We provided a case study to demonstrate how entropy-based detection can be used to identify a TCP-SYN flood attack and how an ACL rule can be implemented to mitigate it. In this chapter, we will assess the effectiveness of the entropy-based detection and ACL rule approach by testing them against different types of (D)DoS attacks. Specifically, we will evaluate the performance of entropy-based (D)DoS detection in Section 5.1, and we will evaluate the performance of the ACL rule in Section 5.2. We will then conclude this chapter in Section 5.3 by answering RQ 5 **(How well does the proposed (D)DoS detection and prevention method perform?)**

5.1 Evaluation of entropy-based (D)DoS detection

In this section, we will evaluate the effectiveness of our entropy-based (D)DoS detection methodology through two approaches. First, we will calculate the detection rate and false positive rate at various threshold values to statistically evaluate the model. Second, we will examine how various attack vectors respond to entropy values. To evaluate the model with different attack vectors, we will reuse the CICDDoS 2019 and ICS Cybersecurity 2018 datasets. These datasets provide us with a range of attack vectors, including TCP SYN flood and ICMP flood from the ICS Cybersecurity 2018 dataset and UDP-flood and UDP-Lag from the CICDDoS 2019 dataset.

5.1.1 Methodology

To statistically evaluate the effectiveness of the entropy-based DDoS detection system, we will calculate the detection rate and false positive rate. The detection rate refers to the rate at which the detection method will correctly identify the (D)DoS traffic and the false positive rate refer to the rate at which the method incorrectly identifies legitimate traffic

as (D)DoS traffic.

In order to determine the detection rate and false positive rate, we follow the following steps:

- Step 1: Collect a dataset of legitimate traffic and (D)DoS traffic.
- Step 2: Create time windows of 1 second and label them as either legitimate or (D)DoS traffic according to the collected dataset.
- Step 3: Calculate the joint entropy for each time window using Equation 4.4.
- Step 4: Compare the joint entropy value for each time window with a threshold value. If the joint entropy value is less than the threshold, label the window as (D)DoS traffic. Otherwise, label it as legitimate traffic.
- Step 5: Calculate the true positive count by counting the number of instances where the joint entropy correctly labels the window as a (D)DoS traffic and the false positive count by counting the number of instances where the joint entropy incorrectly labels the window as a (D)DoS traffic.
- Step 6: Calculate the detection rate as the number of true positives divided by the total number of windows labelled as (D)DoS traffic and the false positive rate as the number of false positives divided by the total number of windows labelled as normal traffic.

After evaluating the model based on the different threshold values, it is relevant to determine how various attack vectors respond to the entropy value. In the second set of experiments, we use four signs to indicate the behaviour of the entropy value for each attribute. A sign of (0) means that the attack exhibits a feature similar to legitimate traffic, and therefore the entropy value does not change. A sign of (-) means that the attribute remains constant throughout the attack, resulting in a decrease in the entropy value. A sign of (+) indicates that the attribute changes continuously during the attack, leading to an increase in the entropy value. A sign of (/) signifies that the attribute is not present in the attack vector. By analyzing the entropy values and their corresponding signs, we can gain insight into how well the entropy-based (D)DoS detection method performs with respect to a different type of (D)DoS attacks. The following section will first describe the detection rate and false positive rate calculation results and then list the results to determine how different attack vectors react to the entropy value.

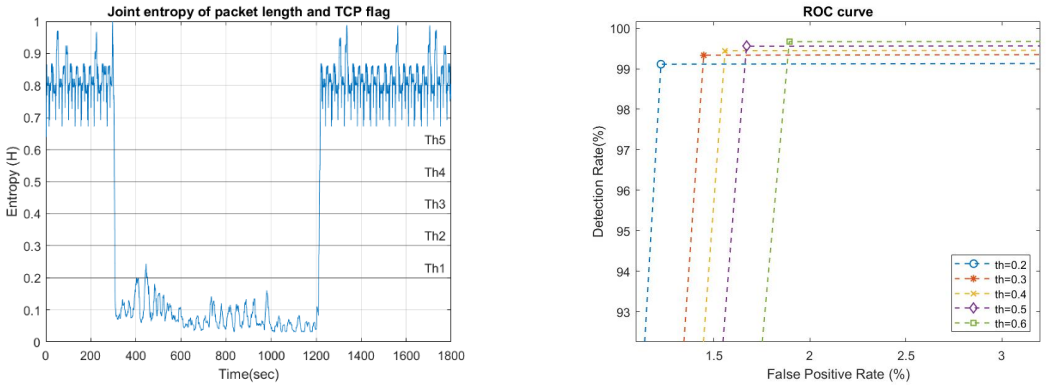
5.1.2 Results

In Section 4.2, we calculated the joint entropy of a TCP SYN flood attack from the ICS Cybersecurity dataset using the packet length and TCP flag attributes to detect (D)DoS

attacks. We recall the output of joint-entropy calculation in Figure 5.1(a). The average joint entropy of legitimate traffic was found to be 0.854, while the average joint entropy during the (D)DoS period (between 300th and 1200th seconds) was 0.781. We compared the decrease in the joint entropy with the threshold value to detect the (D)DoS attack. In this evaluation, we will use the same experiment to calculate the detection rate and false positive rate under five different threshold values (th): {0.2, 0.3, 0.4, 0.5, 0.6}.

The threshold value determines the maximum allowed decrease in joint entropy value. If the threshold value is high, the (D)DoS detection method may be more sensitive to detecting attacks, but it may also have a higher false positive rate. On the other hand, if the threshold value is low, it may result in a lower false positive rate but also a lower detection rate. To illustrate the relationship between the detection and false positive rate, we plot the receiver operating characteristic curve (ROC) for different threshold values as shown in Figure 5.1(b).

The ROC curve is used to assess the accuracy of a model by displaying the relationship between the true and false positives. The ROC curve graphically represents the model's performance across selected thresholds. Scores on the ROC range from 0 to 100, with 100 indicating that the classifier correctly categorizes all labels and 0 indicating that the classifier misclassifies all labels.



(a) Joint entropy of packet length and TCP flag

(b) ROC curve

Figure 5.1: Evaluation of joint entropy of packet length and TCP flag

From the evaluation of the ROC curve, we can see that our entropy-based (D)DoS approach has achieved a high detection rate and low false positive rate when the threshold is optimally defined. By using a threshold value of 0.4, we achieved a detection rate of 99.45%, with a false positive rate as low as 1.56%. With this threshold, we were able to effectively detect the (D)DoS attack.

Table 5.1 show detection rate and false positive rate with different threshold values. From the experimented we found that our approach consistently provided a high detection rate with a low false positive rate. For example, at a threshold of 0.6, the detection rate was

Threshold	Detection rate	False positive rate
0.2	99.11	1.22
0.3	99.37	1.43
0.4	99.45	1.56
0.5	99.52	1.67
0.6	99.66	1.89

Table 5.1: Detection rate and false positive rate score for different values of threshold

99.66%, and the false positive rate was 1.89%. Even when we decreased the acceptance of false positives to 1.22%, the detection rate remained high at 99.11%.

To determine how entropy values can detect different types of DoS attacks, we performed the same entropy calculation on various attack vectors. The attack vectors used for the entropy calculation were the TCP-SYN flood attack, ICMP ping flood attack, UDP flood attack, and UDP-lag attack. For each attack vector, we created 1-second time windows and used a threshold value of 0.4 to assign the decrease or increase in entropy values. As described in the methodology, we used (0),(+) and (-) signs to represent no change, increase, and decrease, respectively in entropy value. Table 5.2 shows the trend of all the attack vectors assessed with our entropy-based detection model.

Attack vector	Entropy						
	Source IP	Destination IP	Protocol.	Source Port	Destination Port	SYN flag	Length
TCP -SYN	+	-	-	+	-	-	-
ICMP	+	-	-	/	/	/	-
UDP	-	-	-	-	-	/	-
UDP-lag	0	0	0	0	0	/	0

Table 5.2: The trend of attributes assessed during the (D)DoS period

From the results, we can observe that the entropy value of packet attributes generally increases or decreases during a DoS attack. In our case, the source IP entropy of the TCP-SYN flood attack and ICMP flood attack increased because the dataset contained forged source IPs during the DoS attack period, while the dataset for the UDP and UDP-lag attacks contained non-forged source IPs during the DoS attack period and did not show an increase in entropy. By analyzing the entropy values of attributes such as destination IP, protocol, and destination port, we can verify that TCP-SYN flood, ICMP flood, and UDP flood attacks generally show a decrease in entropy value, which can be used as a data point for DoS detection. However, we also found a limitation in the entropy calculation's ability to detect attacks like UDP lag. This is because we found that UDP lag appeared as low-volume traffic from the dataset, and the randomness of the attribute fields was similar to that of normal traffic. As a result, the entropy value did not change, leading to the entropy calculation failing to detect the attack.

5.1.3 Concluding remarks

In order to evaluate the effectiveness of our entropy-based (D)DoS detection method, we conducted two experiments. In the first experiment, we calculated the detection rate and false positive rate using different threshold values for the joint-entropy value. The results of this experiment showed that the entropy-based (D)DoS detection method achieved a high detection rate with a low false positive rate. In the second experiment, we analyzed the response of different attack vectors to the entropy calculation. We found that the entropy-based (D)DoS detection method was able to detect TCP-SYN flood attacks, ICMP ping flood attacks, and UDP flood attacks, but it struggled to identify attacks that had similar levels of randomness in their attribute fields as legitimate traffic.

5.2 Evaluation of ACL rule

The ACL rule is created based on the value of the header fields of DoS packets. To evaluate the ACL rule, we used the same datasets that we used to evaluate our entropy-based DoS detection method: CICDDoS 2019 and ICS cybersecurity 2018. In addition, we also obtained a dataset of legitimate traffic from TCPReplay (bigFlows.pcap) [62]. Recall that the ACL rule is only created after the (D)DoS attack is detected by the entropy-based (D)DoS detection method. However, while applying the ACL rule on the Ethernet switch's ACL lookup table, there is a possibility that the Ethernet switch experiences both legitimate traffic and (D)DoS traffic. For this reason it is relevant to evaluate the ACL rule in both legitimate traffic and (D)DoS traffic. To accurately evaluate the effectiveness of the ACL rule in these conditions, we combined the (D)DoS dataset with the legitimate dataset to create a dataset containing both types of traffic.

5.2.1 Methodology

In the previous chapter, we described the algorithm we use to create an ACL rule. This rule is generated after our entropy-based detection methodology has identified a (D)DoS attack. To create the rule, we collect the packets involved in the attack and generate a summary of their header field values. This summary is then converted into an ACL rule. In this evaluation, we will apply the ACL rule to a dataset to test its effectiveness. To measure the performance of the ACL rule, we will consider four types of classifications as follow:

- True positive (TP): The count of malicious packets that are dropped after applying the ACL rule.
- False negative (FN): The count of malicious packets that are allowed after applying the ACL rule

- True negative (TN): The count of legitimate packets that are allowed after applying the ACL rule.
- False positive (FP): The count of legitimate packets that are dropped after applying the ACL rule.

The results can be assessed using the above four classifications. Based on the values, the false positive rate (FPR), the true positive rate (TPR), the precision (PPV) and the accuracy (ACC) are calculated for each attack vector. Equation 5.1 to 5.4 shows the calculation for each classification, respectively.

$$FPR = \frac{FP}{FP + TN} \tag{5.1}$$

$$TPR = \frac{TP}{TP + FN} \tag{5.2}$$

$$PPV = \frac{TP}{FP + TP} \tag{5.3}$$

$$ACC = \frac{TP + TN}{TP + TN + FN + FP} \tag{5.4}$$

The choice of evaluation metrics (such as FPR, TPR, ACC, and PPV) depends on the specific context and goals of the study. In our case, we have found precision to be the most useful metric. Precision measures the percentage of relevant occurrences among all recovered results, and is defined in our study as the proportion of (D)DoS packets among all packets discarded by the ACL rule.

Precision is especially important when false negatives are more preferable than false positives. In our context, it is more acceptable for the ACL rule to allow (D)DoS packets through than to risk discarding legitimate packets. This is because dropping legitimate packets can have negative impacts on the overall system. Therefore, we will use the precision value to evaluate the effectiveness of the ACL rule.

As the FPR, TPR, ACC, and PPV are supposed to be counted at the receiver end. For the evaluation, we will use two types of traffic: packets resulting from (D)DoS captures and packets resulting from legitimate captures.

In order to determine the FPR, TPR, ACC, and PPV, we take the following steps:

- Step 1: Collect a dataset of legitimate traffic and (D)DoS traffic.
- Step 2: Create an ACL rule based on the identified header fields as described in Table 4.4 (For e.g., using the protocol and TCP flag field).
- Step 3: Apply the ACL rule on the (D)DoS traffic and count the number of packets filtered by the rule. (For e.g., using MATLAB script `sum(protocol == 06 && Flag`

`== 'TCP');` to count the number of TCP SYN packets). The count represents the true positive count.

- Step 4: Apply the same ACL filter rule on the legitimate traffic. This time the count represents the false positive count. This is because the ACL rule is intended to filter out only (D)DoS traffic.
- Step 5: Use Equations 5.1 to 5.4 to calculate FPR, TPR, PPV, and ACC.

5.2.2 Results

In Chapter 4, we presented the methodology of ACL rule generation. We discussed the header fields used for the ACL rule and evaluated the ACL rule for the TCP-SYN flood attack. In this section, we will determine the performance by analysing the ACL rule on seven different datasets. All the dataset contains the traces of the (D)DoS attack and legitimate packets. It should be noted that the evaluation of the ACL rule is only performed on publicly available datasets. In the actual scenario, the analysis of the ACL rule will defer based on the rule generated by analysing the (D)DoS attacks. In this section, we will first discuss the results of the ACL rule when applied to the (D)DoS traffic and the legitimate traffic. After that, we will count the FPR, TPR, PPV, and ACC values and specify the header fields which are used for each evaluation.

- **Evaluation on (D)DoS traffic**: The ACL rule is created based on the information provided in the (D)DoS summary. This summary includes the header fields of the (D)DoS packets such as protocol and destination port fields. This summary has sufficient information to detect the (D)DoS packets. For example, considered the UDP flood attack and the ACL rule created based on the protocol field and the destination port. We recall from Section 2.2.1 that the hacker usually sends many UDP packets in a short time period to a particular destination port to consume the target's bandwidth. The ACL rule, which contains the protocol value as UDP and the destination port as the value of the targeted destination port, has sufficient information for identifying and dropping the UDP flood packets. Therefore the false negative value is zero. Similarly, out of seven ACL rule evaluations, all produced a zero false negative count. Subsequently, every evaluation has a true positive rate of 100%.
- **Evaluation on legitimate traffic**: The ACL rule on legitimate traffic showed an impact on the true negative and false positive count of the evaluation. As discussed in the previous section, the true negative count is the count of the legitimate packets allowed by the ACL rule, and the false positive count is the count of the legitimate packets dropped by the ACL rule. In our evaluation, we observed that the ACL rule drops the legitimate packet with the same header field as the (D)DoS packets.

For example, an ACL rule created to drop the UDP packets with a certain port number will essentially drop all the UDP packets to that port. This is one of the main limitations of our (D)DoS mitigation strategy.

Table 5.3 shows the results of evaluating the ACL rule on seven different datasets. The table includes the counts of TP, TN, FP and FN for each evaluation, as well as the header fields used to create the ACL rule. The results indicate that the ACL rule based on the ICMP protocol and type field is more precise than the other ACL rules created for the TCP and UDP protocols. This is because the proportion of legitimate traffic captured from TCPReplay (bigFlows.pcap) [62] that uses the ICMP protocol is only 0.74%, so the ACL rule only affects ICMP echo request packets and does not impact other legitimate traffic. IDs 3 and 4 in the table represent the evaluation of the ACL rule for TCP SYN flood attacks. The ACL rule for these attacks is based on the protocol and flag values. The precision of these rules is 98.31% and 98.27%, respectively, because they only drop TCP packets with the SYN flag. It's important to note that blocking TCP-SYN packets to the targeted device will also involve dropping all TCP connections to the specific target port. The least effective ACL rule was found to be for UDP flood attacks, represented by IDs 5-7. This is because the ACL rule only has information about the protocol field, so it significantly impacts legitimate traffic that uses the UDP protocol.

ID	Protocol	TP	TN	FP	FN	FPR (%)	TPR (%)	ACC (%)	PPV (%)	ACL rule
1	ICMP	961180	786625	4990	0	0.63	100	99.72	99.48	{protocol} {type}
2	ICMP	812000	786625	4990	0	0.63	100	99.38	99.63	{protocol} {type}
3	TCP	1932210	758364	33251	0	4.2	100	98.78	98.31	{protocol} {flag}
4	TCP	1880524	758364	33251	0	4.2	100	98.76	98.27	{protocol} {flag}
5	UDP	316905	637660	153955	0	19.45	100	86.11	67.3	{protocol}
6	UDP	306975	637660	153955	0	19.45	100	85.99	66.6	{protocol}
7	UDP	304836	637660	153955	0	19.45	100	85.96	66.44	{protocol}

Table 5.3: The outcomes of ACL rule categorized by the TP, TN, FP and FN count

5.2.3 Concluding remarks

The section aimed to evaluate the ACL rule. We achieved this goal by counting the false positive rate (FPR), true positive rate (TPR), precision (PPV) and accuracy (ACC) of the ACL rule when applied to the (D)DoS traffic and legitimate traffic. Our evaluation of

the ACL rule showed that it was effective in mitigating (D)DoS traffic, as there were no false negatives. However, it also resulted in the dropping of legitimate packets with the same header fields as (D)DoS packets, leading to an increase in false positives.

5.3 Conclusion

The goal of this chapter was to answer RQ 5 (**How well does the proposed (D)DoS detection and prevention method perform?**). To answer this question, we divided the research question into two parts. Firstly, We evaluate the outcome of (D)DoS detection. It is done by assessing the performance of entropy-based (D)DoS detection on eight different types of (D)DoS attacks. The entropy value of each attack vector was categorized by the change in the entropy value in their header fields. From the results, we concluded that entropy-based detection could detect the (D)DoS attacks, which showed the randomness in their header fields during the (D)DoS attack period. Secondly, a separate set of evaluations was conducted to assess the ACL rule. In all evaluations, the ACL dropped all the (D)DoS packets because the ACL rule was created based on the header fields of (D)DoS packets. But on the other hand, the ACL rule also dropped legitimate packets with the same header field as that of (D)DoS packets. From the finding, we conclude that the (D)DoS mitigation using the ACL rule is possible; however, to increase the efficiency of the (D)DoS mitigation, it is relevant to create an ACL rule based on as many header fields of (D)DoS packets as possible.

Chapter 6

Conclusions and Future Work

6.1 Conclusion

This thesis addresses the detection and prevention mechanism of (D)DoS attacks in automotive Ethernet switches. To achieve this goal, we identified research questions based on the challenges which are faced by today's automotive industries as they trend towards connected and autonomous vehicles. In the following part of the section, we will conclude the thesis by answering each of the research questions.

RQ 1: What are the potential entry points for hackers to gain access to an in-vehicle network?

We answer this research question by performing a literature review on know entry points and identifying components which connect the outside world with in-vehicle networks. We analysed three cases based on the location of the hacker. These cases are long-range wireless access, short-range wireless access and physical access. For each case, we list the entry points and associated attack outcomes. However, the exciting research does not address the risk assessment for each of these entry points.

RQ 2: What are the Ethernet Switch's capabilities for mitigating (D)DoS attacks ?

The goal of this research was to understand the capabilities of Ethernet switches for detecting and preventing (D)DoS attacks. To achieve this, we first reviewed the forwarding engine of Ethernet switches and then analyzed known technologies for detecting these types of attacks. We also conducted a background study on (D)DoS attacks and identified the various stages at which they can be detected and prevented. This allowed us to understand the potential of Ethernet switches in detecting and mitigating these attacks.

RQ 3: How to replicate the Ethernet switch's lookup behaviour in the simulation environment?

To investigate the lookup behaviour of Ethernet switches, we have designed a model based on existing Ethernet switch lookup capabilities. The design comprises three main components. The first component is the parsing logic, which demonstrates how the header fields are extracted from Ethernet frames. The second component is the key generation process, in which a search key is created based on the extracted header fields. The final component is the ACL lookup mechanism, which searches for a match between the generated search key and an ACL rule, and then performs the action associated with that rule. This model serves as the foundation for our (D)DoS detection and mitigation framework.

RQ 4: How to identify known (D)DoS attacks and define ACL rules for Ethernet switch ?

To investigate the effectiveness of our proposed method for detecting and mitigating (D)DoS attacks, we conducted a study that consisted of two parts. In the first part, we analyzed the individual and joint entropy of various attributes in (D)DoS traffic and found that joint entropy was more effective at detecting (D)DoS attacks. In the second part, we defined an ACL rule based on the header fields of (D)DoS packets and applied it to both (D)DoS and legitimate traffic. The results showed that the ACL rule was able to effectively mitigate (D)DoS traffic but also dropped legitimate traffic with the same header fields as (D)DoS packets. To improve the effectiveness of the ACL rule, it is important to carefully consider the header fields used by the rule in order to minimize the impact on legitimate traffic while targeting (D)DoS traffic.

RQ 5: How well does the proposed (D)DoS detection and prevention method perform?

We conducted two experiments to evaluate the effectiveness of our entropy-based (D)DoS detection method. The first experiment showed that the method had a high detection rate and a low false positive rate using different threshold values for joint entropy. The second experiment showed that the method could identify TCP-SYN flood attacks, ICMP ping flood attacks, and UDP flood attacks but did not identify attack vectors with the same level of randomness as legitimate traffic. To evaluate the ACL rule, we measured the FPR, TPR, PPV, and ACC of the rule when applied to (D)DoS and legitimate traffic. The ACL rule effectively mitigated (D)DoS traffic but also dropped legitimate traffic with the same header fields as (D)DoS packets, leading to false positives.

Overall, the goal of this thesis was to provide a (D)DoS detection and prevention framework in automotive Ethernet switches. To do this, We identified several research questions related to potential entry points for hackers, the capabilities of Ethernet switches for mitigating (D)DoS attacks, how to replicate Ethernet switch behaviour in a simulation environment, how to identify known (D)DoS attacks and define ACL rules for Ethernet switches, and the outcomes of (D)DoS detection and prevention efforts. We conducted a literature review, designed a model based on Ethernet switch capabilities, and conducted experiments to evaluate the effectiveness of (D)DoS detection and prevention method.

6.2 Future work

Although our (D)DoS detection and mitigation framework is able to detect and mitigate (D)DoS attacks, there is scope for future work in the following areas:

Design of Ethernet switch model

Our current design implementation is created to demonstrate the ACL lookup mechanism of the Ethernet switch. It includes the parsing logic, key generation mechanism and ACL lookup mechanism. However, this design does not consider the limitation of the number of rules which can be assigned to an ACL lookup table. In a real scenario, the number of rules which can be defined in the ACL lookup table depends on the length of the search key and the number of rules. A standard Ethernet switch can store limited rules in the ACL lookup table [10]. Future work should address the limited number of ACL rules and identify effective ways to utilize them.

Entropy-based (D)DoS detection

The entropy of the network's traffic depends on the randomness of the type of packets in a given time window. During a (D)DoS attack, the randomness of the traffic usually decreases or increases compared to legitimate traffic behaviour. However, (D)DoS attacks based on the fragmentation attack, like the Teardrop attack and ICMP ping of death attack, may not show a significant change in the entropy value. This is because fragmentation-based (D)DoS attacks usually require only one or two packets, and the randomness in the traffic would not change extensively as compared to flooding attacks like TCP-SYN flood and UDP flood. For future research, attack vectors such as Teardrop attacks and ICMP ping of death attacks should be considered for the evaluation.

Evaluation of ACL rule

We evaluated the ACL rule by implementing it on (D)DoS and legitimate datasets. The evaluation showed that the ACL rule effectively dropped (D)DoS packets but also dropped legitimate packets with the same header fields. This happened because our approach excluded the source IP address from the ACL rule, causing packets with legitimate source IPs to be affected. A potential solution is to group malicious source IPs and apply the ACL rule to them. Another limitation of our evaluation is that it was conducted in a simulation environment. In the real world, the ACL rule must be created in OS-specific language and evaluated using real hardware switches. Future research should implement the provided (D)DoS detection and mitigation algorithm in real hardware switches to evaluate the proposed framework.

Bibliography

[1] A. Pretschner, M. Broy, I. H. Kruger, and T. Stauner, "Software engineering for automotive systems: A roadmap," in *2007 Future of Software Engineering*, ser. FOSE '07. USA: IEEE Computer Society, 2007, p. 55–71. [Online]. Available: https://doi.org/10.1109/FOSE.2007.22 1

[2] J. E. Siegel, D. C. Erb, and S. E. Sarma, "A survey of the connected vehicle landscape—architectures, enabling technologies, applications, and development areas," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, pp. 2391–2406, 2018. 1

[3] G. Abdelkader, K. Elgazzar, and A. Khamis, "Connected vehicles: Technology review, state of the art, challenges and opportunities," *Sensors*, vol. 21, no. 22, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/22/7712 1

[4] D. M. Von Paul Gao, Hans-Werner Kaas and D. Wee, "Automotive revolution – perspective towards 2030," https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/disruptive-trends-that-will-transform-the-auto-industry/de-DE, accessed: 2022-09-10. 1

[5] "Road traffic injuries," https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries, accessed: 2022-09-10. 2

[6] M. Dibaei, X. Zheng, Y. Xia, X. Xu, A. Jolfaei, A. K. Bashir, U. Tariq, D. Yu, and A. V. Vasilakos, "Investigating the prospect of leveraging blockchain and machine learning to secure vehicular networks: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 683–700, 2022. 2

[7] C. Miller and C. Valasek., "Remote exploitation of an unaltered passenger vehicle," https://illmatics.com/Remote%20Car%20Hacking.pdf, accessed: 2022-09-10. 2, 8, 9

[8] C. Yan, W. Xu, and J. Liu, "Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle," *Def Con*, vol. 24, no. 8, p. 109, 2016. 2

[9] A. Greenberg, "A new wireless hack can unlock 100 million volkswagens," https://www.wired.com/2016/08/oh-good-new-hack-can-unlock-100-million-volkswagens/, accessed: 2022-09-10. 2

[10] Huawei, "S7700 and s9700 v200r007(c00c10) configuration guide - security," https://support.huawei.com/enterprise/en/doc/EDOC1000069539?section=j004, accessed: 2022-09-10. 3, 19, 21, 61

[11] P. S. Oruganti, M. Appel, and Q. Ahmed, "Hardware-in-loop based automotive embedded systems cybersecurity evaluation testbed," in *Proceedings of the ACM Workshop on Automotive Cybersecurity*, ser. AutoSec '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 41–44. [Online]. Available: https://doi.org/10.1145/3309171.3309173 8

[12] R. Piqueras Jover, "Security attacks against the availability of lte mobility networks: Overview and research directions," in *2013 16th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, 2013, pp. 1–9. 8

[13] Y. Zhang, B. Ge, X. Li, B. Shi, and B. Li, "Controlling a car through obd injection," *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*, pp. 26–29, 2016. 8

[14] D. S. Fowler, J. Bryans, S. A. Shaikh, and P. Wooderson, "Fuzz testing for automotive cyber-security," *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 239–246, 2018. 8

[15] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *Proceedings of the 20th USENIX Conference on Security*, ser. SEC'11. USA: USENIX Association, 2011, p. 6. 8

[16] H. Wen, Q. A. Chen, and Z. Lin, "Plug-n-pwned: Comprehensive vulnerability analysis of obd-ii dongles as a new over-the-air attack surface in automotive iot," in *Proceedings of the 29th USENIX Conference on Security Symposium*, ser. SEC'20. USA: USENIX Association, 2020. 8

[17] N. Nissim, R. Yahalom, and Y. Elovici, "Usb-based attacks," *Computers Security*, vol. 70, pp. 675–688, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404817301578 9

[18] G. Loukas, T. Vuong, R. Heartfield, G. Sakellari, Y. Yoon, and D. Gan, "Cloud-based cyber-physical intrusion detection for vehicles using deep learning," *IEEE Access*, vol. 6, pp. 3491–3508, 2018. 9

[19] E. F. M. Josephlal and S. Adepu, "Vulnerability analysis of an automotive infotainment system's wifi capability," *2019 IEEE 19th International Symposium on High Assurance Systems Engineering (HASE)*, pp. 241–246, 2019. 9

[20] P. Gullberg, "Denial of service attack on bluetooth low energy," 09 2016. [Online]. Available: http://dx.doi.org/10.13140/RG.2.2.12059.26407 9

[21] A. Francillon, B. Danev, and S. Capkun, "Relay attacks on passive keyless entry and start systems in modern cars." *IACR Cryptology ePrint Archive*, vol. 2010, p. 332, 01 2010. 9

[22] K. Lounis and M. Zulkernine, "Exploiting race condition for wi-fi denial of service attacks," in *13th International Conference on Security of Information and Networks*, ser. SIN 2020. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3433174.3433584 9

[23] A. M. C. P. M. S. Thomas Eisenbarth, Timo Kasper and M. T. Shalmani., "Physical cryptanalysis of keeloq code hopping applications," https://eprint.iacr.org/2008/058.pdf, accessed: 2022-09-10. 9

[24] I. Rouf, R. D. Miller, H. A. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar, "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study," in *USENIX Security Symposium*, 2010. 9

[25] K. Matheus and T. Königseder, *Automotive Ethernet*, 2nd ed. Cambridge University Press, 2017. 10

[26] Cloudflare, "Ping (icmp) flood ddos attack," https://www.cloudflare.com/learning/ddos/ping-icmp-flood-ddos-attack/, accessed: 2022-10-26. 11

[27] ——, "Udp flood attack," https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/, accessed: 2022-10-26. 11

[28] ——, "Smurf ddos attack," https://www.cloudflare.com/learning/ddos/smurf-ddos-attack/, accessed: 2022-10-26. 12

[29] Radware, "Fraggle attack," https://www.radware.com/security/ddos-knowledge-center/ddospedia/fraggle-attack/, accessed: 2022-10-26. 12

[30] Cloudflare, "Syn flood attack," https://www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/, accessed: 2022-10-26. 12

[31] ——, "Ping of death ddos attack," https://www.cloudflare.com/learning/ddos/ping-of-death-ddos-attack/, accessed: 2022-10-26. 12

[32] C. Maple, M. Bradbury, A. T. Le, and K. Ghirardello, "A connected and autonomous vehicle reference architecture for attack surface analysis," *Applied Sciences*, vol. 9, no. 23, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/23/5101 13

[33] S. Traer and P. Bednar, "Motives behind ddos attacks," in *Digital Transformation and Human Behavior*, C. Metallo, M. Ferrara, A. Lazazzara, and S. Za, Eds. Cham: Springer International Publishing, 2021, pp. 135–147. 13

[34] Cloudfare, "Famous ddos attacks — the largest ddos attacks of all time," https://www.cloudflare.com/learning/ddos/famous-ddos-attacks/, accessed: 2022-09-10. 13

[35] L. H. Newman, "Menacing malware shows the dangers of industrial system sabotage," https://www.wired.com/story/triton-malware-dangers-industrial-system-sabotage/?CNDID=50121752., accessed: 2022-09-10. 13

[36] T. Guardian, "Russia accused of unleashing cyberwar to disable estonia," https://www.theguardian.com/world/2007/may/17/topstories3.russia, accessed: 2022-09-10. 13

[37] R. Hayton, "Why would someone hack your car?" https://www.trustonic.com/opinion/who-would-want-to-hack-your-car/, accessed: 2022-10-26. 13

[38] Cloudfare, "What is a ddos booter/ip stresser? — ddos attack tools," https://www.cloudflare.com/learning/ddos/ddos-attack-tools/ddos-booter-ip-stresser/. 13

[39] N. Kshetri, "The simple economics of cybercrimes," *IEEE Security Privacy*, vol. 4, no. 1, pp. 33–39, 2006. 14

[40] F. C. Freiling, T. Holz, and G. Wicherski, "Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks," in *European Symposium on Research in Computer Security*. Springer, 2005, pp. 319–335. 15

[41] G. R. Faulhaber, "Economics of net neutrality: A review," *Communications & Convergence Review*, vol. 3, no. 1, pp. 53–64, 2011. [Online]. Available: https://ssrn.com/abstract=1894286 15

[42] M. Geva, A. Herzberg, and Y. Gev, "Bandwidth distributed denial of service: Attacks and defenses," *IEEE Security & Privacy*, vol. 12, no. 1, pp. 54–61, 2013. 18

[43] A. Ayorinde, "Comparative analysis of various denials of service (dos) attack mitigation techniques," vol. Vol. 8 No.04 Jul-Aug 2019, pp. 163–167, 07 2019. 19

[44] P. Ioulianou, V. Vasilakis, I. Moscholios, and M. Logothetis, "A signature-based intrusion detection system for the internet of things," *Information and Communication Technology Form*, 2018. 20

[45] C. Buragohain, M. J. Kalita, S. Singh, and D. K. Bhattacharyya, "Anomaly based ddos attack detection," *International Journal of Computer Applications*, vol. 123, no. 17, 2015. 20

[46] F. Luo, B. Wang, Z. Fang, Z. Yang, and Y. Jiang, "Security analysis of the tsn backbone architecture and anomaly detection system design based on ieee 802.1qci," *Security and Communication Networks*, vol. 2021, p. 6902138, Sep 2021. [Online]. Available: https://doi.org/10.1155/2021/6902138 21

[47] D. Medhi and K. Ramasamy, "Chapter 15 - ip packet filtering and classification," in *Network Routing (Second Edition)*, second edition ed., ser. The Morgan Kaufmann Series in Networking, D. Medhi and K. Ramasamy, Eds. Boston: Morgan Kaufmann, 2018, pp. 500–547. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128007372000181 21

[48] Z. Wang, H. Che, J. Cao, and J. Wang, "A tcam-based solution for integrated traffic anomaly detection and policy filtering," *Computer Communications*, vol. 32, no. 17, pp. 1893–1901, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366409002102 21

[49] R. Amin, I. Pali, and V. Sureshkumar, "Software-defined network enabled vehicle to vehicle secured data transmission protocol in vanets," *Journal of Information Security and Applications*, vol. 58, p. 102729, 2021. 21

[50] O. S. Al-Heety, Z. Zakaria, M. Ismail, M. M. Shakir, S. Alani, and H. Alsariera, "A comprehensive survey: Benefits, services, recent works, challenges, security, and use cases for sdn-vanet," *IEEE Access*, vol. 8, pp. 91 028–91 047, 2020. 21

[51] T. Häckel, A. Schmidt, P. Meyer, F. Korf, and T. C. Schmidt, "Strategies for integrating control flows in software-defined in-vehicle networks and their impact on network security," in *2020 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2020, pp. 1–8. 21

[52] C.-H. Y. Y.-C. W. Ying-Dar Lin, Po-Ching Lin and Y.-C. Lai, "An extended sdn architecture for network function virtualization with a case study on intrusion prevention," *Int'l Journal on IEEE Network*, vol. 29, 2015. 21

[53] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "Nice: Network intrusion detection and countermeasure selection in virtual network systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 99, p. 1, 07 2013. 21

[54] Q. Yan and F. Yu, "Distributed denial of service attacks in software-defined networking with cloud computing," *Comm. Mag.*, vol. 53, no. 4, p. 52–59, apr 2015. [Online]. Available: https://doi.org/10.1109/MCOM.2015.7081075 21

[55] M. A. Aladaileh, M. Anbar, A. J. Hintaw, I. H. Hasbullah, A. A. Bahashwan, and S. Al-Sarawi, "Renyi joint entropy-based dynamic threshold approach to detect ddos attacks against sdn controller with various traffic rates," *Applied Sciences*, vol. 12, no. 12, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/12/6127 21

[56] SJA1110 UM, "Unpublished confidential document," 2021. 23, 24

[57] K. Kalkan, G. Gür, and F. Alagöz, "Defense mechanisms against ddos attacks in sdn environment," *IEEE Communications Magazine*, vol. 55, pp. 175–179, 2017. 32

[58] I. Frazão, P. H. Abreu, T. Cruz, H. Araújo, and P. Simões, "Denial of service attacks: Detecting the frailties of machine learning algorithms in the classification process," in *Springer series on Security and Cryptology*, 2018. 33, 36

[59] S. H. Iman Sharafaldin, Arash Habibi Lashkari and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *IEEE 53rd International Carnahan Conference on Security Technology*, 2019. 33

[60] M. Bellaiche and J.-C. Gregoire, "Syn flooding attack detection based on entropy computing," in *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, 2009, pp. 1–6. 34

[61] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to ddos attack detection and response," *Proceedings DARPA Information Survivability Conference and Exposition*, vol. 1, pp. 303–314 vol.1, 2003. 34

[62] tcpreplay, "Sample captures," https://tcpreplay.appneta.com/wiki/captures.html, accessed: 2022-09-10. 54, 57