To: **King Fahd university of Petroleum & Minerals**

Deanship of Student Affairs

Vice Deanship for Student Excellence & Success

Undergraduate Research Office (URO)

Undergraduate Student Research Grant (Uxplore)-Term 232

Final Report

Student's Name	Basil Alashqar	Advisor' Name	Dr. Waleed Al-Gobi
ID	202045700	Research Center	IRC for Intelligent Secure Systems
Department	Computer Engineering	Research Topic	IP Reputation (In-network Detection of ICMPv6 DDoS Attacks)
Level	Junior		

Date:

15/06/2024

Advisor's Signature:

In-network Detection of ICMPv6 DDoS Attacks

Basil Alashqar

IRC for Intelligent Secure Systems
KFUPM
Saudi Arabia
alashqar.ba@hotmail.com

ABSTRACT

This report details the implementation and evaluation of a P4 program aimed at detecting and mitigating ICMPv6 Neighbor Discovery Protocol (NDP) attacks using Bloom filters, Count-Min Sketch (CMS), and timestamp-based analysis. Given the limitations of traditional defense mechanisms like rate limiting and signature-based detection, our solution leverages these probabilistic data structures to efficiently track and analyze traffic patterns. The program processes incoming packets, extracts necessary headers, updates counters, and applies defense mechanisms based on detected patterns. To simulate realistic ICMPv6 attack scenarios, the Scapy library was used, incorporating random delays, variable packet bursts, and time-based simulation durations. The experimental setup, configured on a Linux Ubuntu system with a BMV2 switch and multiple virtual hosts, demonstrated the program's effectiveness in significantly reducing malicious traffic throughput and achieving a high percentage of dropped packets, thereby underscoring the potential of programmable switches and probabilistic data structures in enhancing network security against evolving ICMPv6 attacks.

1. INTRODUCTION

The Internet Control Message Protocol for IPv6 (ICMPv6) is an integral part of the IPv6 protocol suite, primarily used for error messaging and diagnostic functions, such as the well-known "ping" command. However, ICMPv6 is also susceptible to various network attacks that can disrupt normal network operations. One of the critical functionalities of ICMPv6 is the Neighbor Discovery Protocol (NDP), which is essential for various network operations including address autoconfiguration, address resolution, and neighbor unreachability detection. Attacks exploiting NDP, often referred to as ICMPv6 Neighbor Discovery attacks, can overwhelm network resources, leading to Denial of Service (DoS) conditions. Traditional defense mechanisms against such attacks include rate limiting and signature-based detection systems. Rate limiting involves controlling the rate of ICMPv6 packet flow to prevent excessive traffic, while signature-based systems detect known attack patterns by comparing incoming packets against a database of attack signatures. However, these methods have limitations. Rate limiting can be too restrictive, potentially affecting legitimate traffic, and signature-based systems struggle to detect new or evolving attack patterns that are not yet cataloged.

2. RELATED WORK

Traditional defense mechanisms, such as rate limiting and signature-based detection, have been employed to mitigate ICMPv6 attacks. However, these methods are often insufficient in dealing with sophisticated and evolving attack patterns. Rate limiting can be overly restrictive, impacting legitimate traffic, while signature-based detection requires constant updates and can be evaded by attackers using new techniques. The need for more adaptive and efficient solutions has led to the exploration of probabilistic data structures and programmable switches as promising alternatives.

3. BACKGROUND AND MOTIVATION

The P4 language is designed for programming protocol-independent packet processors. It allows network operators to define how packets are processed by the data plane of network devices, such as switches and routers. This programmability enables the implementation of advanced packet processing techniques directly in the network hardware, offering high performance and flexibility. In this project, a P4 program is developed to detect and mitigate ICMPv6 Neighbor Discovery attacks using Bloom filters, CMS, and timestamps. The program processes incoming packets, extracts necessary headers, updates counters, and applies defense mechanisms based on detected packet patterns. By tracking unique packet elements and their frequencies, along with timestamps of their occurrences, the program can identify potential attack patterns and take appropriate actions, such as dropping malicious packets, to protect the network.

4. METHODOLOGY

4.1 Threat Model

The threat model for this project focuses on ICMPv6 Neighbor Discovery Protocol (NDP) attacks, which can severely disrupt network operations by flooding the network with malicious ICMPv6 packets. These attacks exploit the essential functions of NDP, such as address resolution and neighbor unreachability detection, leading to resource exhaustion and denial of service. The primary goal of the defense mechanisms designed in this project is to detect and mitigate these ICMPv6 NDP attacks in real-time, minimizing their impact on network performance.

4.2 Defense Mechanism

To defend against ICMPv6 Neighbor Discovery Protocol (NDP) attacks, several probabilistic data structures and algorithms were implemented in P4. Bloom filters are used to track unique packet elements with a small probability of false positives, updating entries whenever a new packet element is detected. Count-Min Sketch (CMS) is employed to estimate the frequency of elements in a data stream, utilizing multiple CMS registers to minimize the probability of overestimation due to hash collisions. Additionally, timestamp registers are used for timestamp analysis, tracking the last seen time of packet elements to help identify patterns of rapid, repetitive traffic indicative of flooding attacks.

4.3 Proposed Algorithm

The defense mechanism is structured into several key algorithms:

4.3.1 Jenkins Hash Function

The Jenkins hash function is a popular non-cryptographic hash function designed to distribute data uniformly. It is particularly known for its efficiency and simplicity, making it suitable for various applications, including hash tables and probabilistic data structures like Bloom filters.

The function operates on a key (such as a string or an array of bytes) and produces a hash value. The key is processed byte-by-byte through a series of bitwise operations to produce a final hash value. These operations ensure that even small changes in the input key result in significantly different hash values, which helps in uniformly distributing data.

4.3.1.1 Steps of the Jenkins Hash Function:

1. **Initialization**: Start with an initial hash value (h) set to 0.
2. **Processing Each Byte**: For each byte in the key:
   - Add the byte to the hash value (h).
   - Perform a left bitwise shift of the hash value by 10 bits and add the result to h.
   - Perform a right bitwise shift of the hash value by 6 bits and XOR the result with h.
3. **Final Mixing**: After processing all bytes:
   - Perform a left bitwise shift of the hash value by 3 bits and add the result to h.
   - Perform a right bitwise shift of the hash value by 11 bits and XOR the result with h.
   - Perform a left bitwise shift of the hash value by 15 bits and add the result to h.
4. **Return the Hash Value**: The final value of h is the hash value for the given key.

This series of operations ensures a good mix of the input data, resulting in a well-distributed hash value.

4.3.2 Packet Counting and Timestamp Update

Packet counting and timestamp updates are essential for tracking the number of processed packets and their arrival times. This information helps in implementing defense mechanisms that rely on packet rates and timing.

4.3.2.1 Steps of Packet Coutning and Timestamp Updates

1. **Initialization**: For each incoming packet, get the current timestamp.
2. **Increment Packet Counter**: Increment the total_packet_counter_reg to count the packet.
3. **Check Time Since Last Packet**: Calculate the time difference between the current time and the last_packet_time.
   - If this time difference exceeds 100 seconds, set the bypass_defense flag to 1.
   - Otherwise, set the bypass_defense flag to 0.
4. **Update Last Packet Time**: Update the last_packet_time with the current time.

By maintaining these counts and timestamps, the system can decide when to bypass defense mechanisms and when to apply them based on packet rates and timing.

4.3.3 ICMPv6 Packet Handling

ICMPv6 packet handling involves specific steps to process these packets, increment counters, and update data structures such as Bloom filters and CMS.

4.3.3.1 Steps of ICMPv6 Packet Handling

1. **Check Packet Type**: If the packet is an ICMPv6 packet, proceed with handling.
2. **Increment ICMPv6 Counter**: Increment the icmpv6_counter_reg to count the ICMPv6 packet.
3. **Generate Hashes**: Use the Jenkins hash function to generate three hash values for the packet's address.
4. **Update Bloom Filter and CMS**: For each generated hash value:
   - Set the corresponding Bloom filter entry to 1.
   - Increment the corresponding CMS entry.
   - Update the corresponding timestamp entry with the current time.

These steps ensure that ICMPv6 packets are processed, and their information is recorded in the Bloom filter and CMS for further analysis.

4.3.4 Bloom Filter Check

The Bloom filter check determines whether an element has been seen before by checking the corresponding entries in the Bloom filter. If all entries are zero, the element is new; otherwise, additional checks are performed.

4.3.4.1 Steps of the Bloom Filter Check

1. **Check Bloom Filter Entries**: For each of the three hash values generated by the Jenkins hash function, check the corresponding Bloom filter entries.
   - If all three entries are zero, set them to one, indicating a new element.
   - Otherwise, proceed to check the timestamps.
2. **Check Timestamps**: Calculate the minimum time difference between the current time and the last seen times for the three hash values.
   - If this time difference is less than a defined threshold, set the should_drop flag to 1.

This process helps identify whether an element is new or has been seen recently, aiding in the decision to drop or forward the packet.

4.3.5 Count-Min Sketch and Frequency Check

1. **Get Minimum Count**: For each of the three hash values generated by the Jenkins hash function, get the counts from the corresponding CMS registers.
    - Calculate the minimum count among these values.
2. **Check Frequency**: Compare the minimum count with a predefined threshold.
    - If the minimum count exceeds the threshold, set the should_drop flag to 1.
    - Otherwise, increment the counts in the CMS registers for the hash values.

This method provides an approximate frequency count, helping identify elements that appear frequently and may indicate malicious activity.

4.3.6 Drop Decision

The drop decision is the final step in determining whether to drop or forward a packet based on the should_drop flag set during previous checks.

- If the flag is set to 1, drop the packet.
- Otherwise, forward the packet.

4.4 Implementation Details

The P4 program is implemented with various components to ensure robust and efficient defense mechanisms against ICMPv6 Neighbor Discovery Protocol (NDP) attacks. It includes several metadata fields such as should_drop, flood_detected, and bypass_defense to control packet processing decisions. Various registers are used to track packet counts, Bloom filter entries, Count-Min Sketch (CMS) counts, and timestamps. By utilizing multiple Bloom filter registers, the probability of false positives is reduced, as each register independently tracks the presence of elements, providing a more accurate representation. Similarly, multiple CMS registers minimize the probability of overestimation due to hash collisions; by taking the minimum value across these registers, the system achieves a more accurate frequency estimate. Employing multiple timestamp registers allows the system to track the timing of elements more accurately, reducing the likelihood of hash collisions affecting the timing information and ensuring that legitimate packets are not mistakenly dropped. The ingress control block is where the core algorithms, such as Bloom filtering, CMS, and timestamp analysis, are implemented. By implementing these detailed steps and utilizing multiple registers, the P4 program provides a robust and efficient defense mechanism against ICMPv6 NDP attacks, ensuring real-time traffic analysis and intelligent decision-making.

5. EVALUATION

5.1 Experiments' Goals

The primary goal of the experiments was to evaluate the effectiveness of the P4 program in detecting and mitigating ICMPv6 Neighbor Discovery Protocol (NDP) attacks. This involved comparing the network performance with and without the defense mechanisms in place.

5.2 SETUP

The experimental setup was configured virtually on a Linux Ubuntu system, utilizing a BMv2 (Behavioral Model version 2) switch to simulate a network environment. This setup consisted of eight hosts, each connected to a distinct port on the BMv2 switch, with each host assigned a unique IPv6 address and MAC address. The topology of the experimental setup is illustrated in Figure 1.

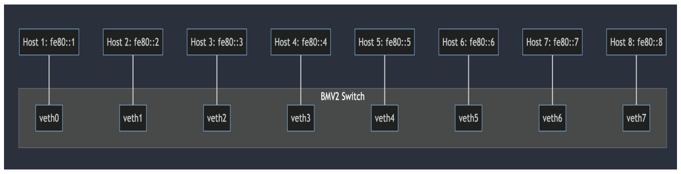

Figure 1. Setup Topology: The topology of the experimental setup showing the BMv2 switch and the connected virtual hosts.

To simulate ICMPv6 attacks, the Scapy library was employed due to the limited availability of real-world datasets for such attacks. Several measures were implemented in the Scapy code to ensure the simulated attacks closely resembled realistic scenarios. Delays between attack packets were randomly set between 0.2 seconds and 2 seconds to mimic the irregular intervals that might occur in actual network attacks. The number of packets sent at one time was randomly chosen to be 1, 2, or 3, adding another layer of realism to the attack simulation. Attacks were simulated based on time durations of 3 minutes, 5 minutes, 7 minutes, and 10 minutes, rather than a specific number of packets. This approach ensures that the attack patterns are closer to what might be observed in real-world scenarios, where attacks are sustained over time rather than defined by a fixed number of packets. Figure 2 shows the setup for the attack simulations.

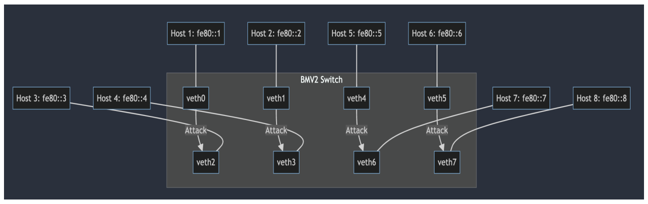

Figure 2. Attacks Test Setup: The setup for the attack simulations showing the attack paths from different hosts.

5.3 EVALUATION METRICS

The performance and effectiveness of the defense mechanisms were evaluated using several metrics: the total number of packets processed during the simulation, the number of packets identified as malicious and dropped by the defense mechanisms, the percentage of total packets that were dropped, and the rate at which packets were processed by the switch (measured in bits per second, or bps).

5.4 RESULTS

The results demonstrated the effectiveness of the defense mechanisms in mitigating ICMPv6 attacks. Table 1 summarizes the tests conducted, showing the number of dropped packets, forwarded packets, total packets, percentage dropped, and throughput consumed by the attack.

Table 1. Tests Results

Attack Scenario	Defense Status	Dropped Packets	Forwarded Packets	Total Packets	Percentage Dropped	Throughput Consumed by the Attack (bps)
H1 Attacks H3 (3 mins)	With Defense	2129	550	2679	79.47%	782.22
H1 Attacks H3 (3 mins)	Without Defense	0	1113	1113	0.00%	1582.93
H2 Attacks H4 (5 mins)	With Defense	3600	880	4480	80.36%	750.93
H2 Attacks H4 (5 mins)	Without Defense	0	1967	1967	0.00%	1678.51
H5 Attacks H7 (7 mins)	With Defense	4979	1221	6200	80.31%	744.23
H5 Attacks H7 (7 mins)	Without Defense	0	2741	2741	0.00%	1670.70
H6 Attacks H8 (10 mins)	With Defense	7252	1649	8901	81.47%	703.57
H6 Attacks H8 (10 mins)	Without Defense	0	3584	3584	0.00%	1529.17

5.4.1 Total Packets

The total number of packets processed is significantly higher when defense mechanisms are active. This is because packets marked as dropped are processed very quickly at the beginning of the defense, allowing new packets to come in rapidly. Without defense, each packet takes more time to process fully, leading to a lower number of total packets being handled in the same duration. This is illustrated in Figure 3.

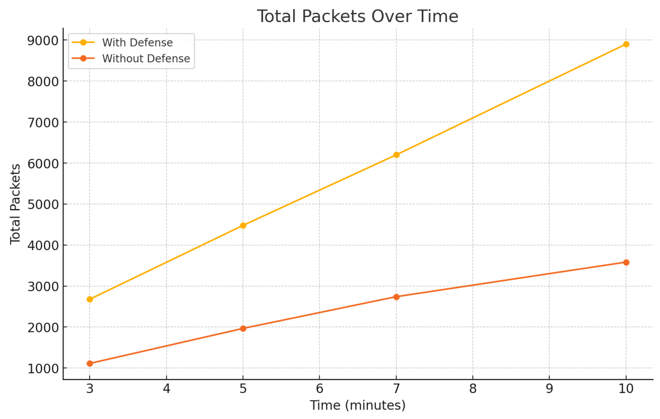

Figure 3. Packets Versus Time: Graph showing the total number of packets processed over time with and without defense mechanisms.

5.4.2 Dropped Packets

With defense, a substantial number of packets were dropped, reflecting the effectiveness of the defense mechanisms in identifying and mitigating the attack traffic. Without defense, no packets were dropped, which shows that all the attack packets were being processed, contributing to the attack's success. This is illustrated in Figure 4.

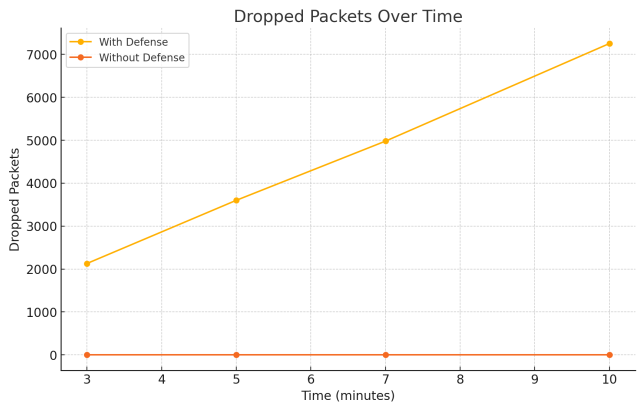

Figure 4. Dropped Packets Over Time: Graph showing the number of dropped packets over time with and without defense mechanisms.

5.4.3 Percentage Dropped

The percentage of dropped packets is a crucial metric that reflects the effectiveness of the defense mechanisms in place during ICMPv6 attacks. When defense mechanisms were active, the consistently high percentage of dropped packets (around 80%) across all attack scenarios indicated that the defense system was effectively identifying and discarding malicious packets. This high drop rate demonstrated the defense mechanism's capability to filter out attack traffic, thereby preventing it from reaching its intended target. By dropping a large portion of the attack packets early in the processing chain, the defense mechanism prevented the network from being overwhelmed. This efficiency is crucial in maintaining the overall stability and performance of the network, as it allows legitimate traffic to be processed without significant delay or interruption.

In contrast, in the absence of defense mechanisms, no packets were dropped. This means that all packets, including malicious ones, were fully processed. The lack of dropped packets indicated that the system was not distinguishing between attack and legitimate traffic, allowing all incoming packets to consume network resources. Without dropping any packets, the network was more vulnerable to being overwhelmed by the attack traffic. This could lead to higher latency, reduced performance, and potential service disruptions as the system struggled to handle the excessive load generated by the attack. A high percentage of dropped packets during an attack helps maintain network stability by ensuring that only a fraction of the incoming traffic is processed and forwarded. This prevents the network from being flooded with malicious traffic, which could otherwise lead to network congestion and potential failures. By dropping a significant portion of the attack traffic, the defense mechanism allows network resources (such as bandwidth and processing power) to be allocated more efficiently. This ensures that legitimate traffic receives the necessary resources for smooth operation, enhancing the overall performance of the network. A high drop rate signifies effective attack mitigation. The defense system quickly identifies and discards malicious packets, reducing the attack's impact on the network. This proactive approach helps in maintaining the quality of service and minimizing the potential damage caused by the attack. While a high drop rate leads to a reduction in throughput, this reduction is a positive outcome during an attack scenario. It indicates that the defense mechanism is successfully preventing a large volume of attack traffic from consuming network resources. The lower throughput reflects the system's ability to prioritize and protect legitimate traffic. This is illustrated in Figure 5.

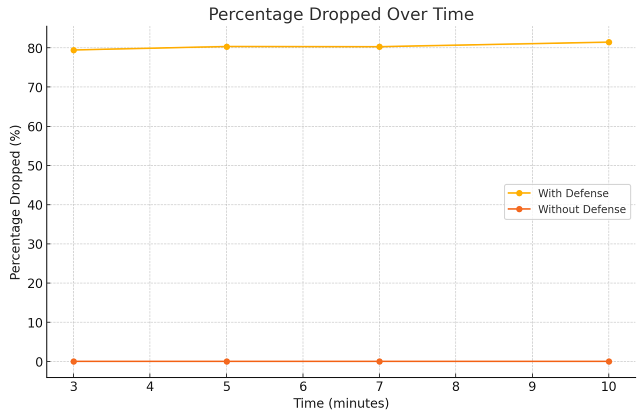

Figure 5. Percentage of Dropped Packets Over Time: Graph showing the percentage of dropped packets over time with and without defense mechanisms.

5.4.4 Throughput

The throughput (in bps) consumed by the whole switch was lower with defense because the system quickly dropped attack packets, resulting in fewer packets being processed and forwarded. Without defense, the throughput consumed by the attack was higher since all packets, including attack packets, were processed and forwarded, leading to a higher data rate. This is illustrated in Figure 6.

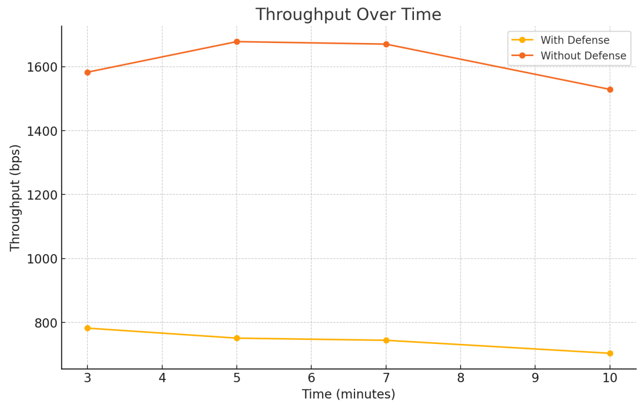

Figure 6. Throughput Consumed By Attack Over Time: Graph showing the throughput consumed by the attack over time with and without defense mechanisms.

5.5 Results Conclusion

The results demonstrated that we successfully employed the BMv2 programmable switch with P4 language to effectively mitigate ICMPv6 attacks. Through our defense mechanisms, we were able to efficiently drop malicious packets, maintain higher total packet handling, and reduce throughput consumed by attack traffic. The high drop rate showcased the effectiveness of our P4-based defense code in filtering out attack traffic early in the processing chain. With defense mechanisms active, the total number of packets processed was significantly higher due to the rapid processing of dropped packets. This is crucial for maintaining network performance during an attack. The throughput consumed by attack traffic was notably lower with defense mechanisms in place, preventing network congestion and ensuring that legitimate traffic received the necessary resources for smooth operation. The results highlighted the importance of implementing robust defense strategies using programmable switches and P4 language, effectively protecting the network from being overwhelmed by attack traffic and maintaining stability and performance under attack conditions. This comprehensive analysis is illustrated in Figure 7.

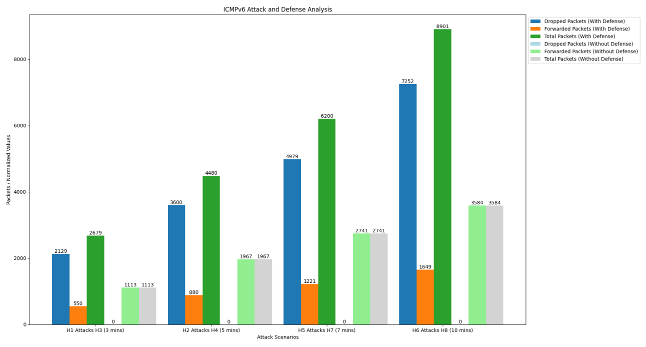

Figure 7. Compressive Graph: A comprehensive graph showing the results of ICMPv6 attack and defense analysis.

6. DISCUSSION

The evaluation results have highlighted several key points regarding the effectiveness of our P4-based defense mechanisms against ICMPv6 attacks. The defense mechanisms demonstrated a significant capability to identify and mitigate malicious traffic. Across various attack scenarios, the system consistently achieved a high percentage of dropped packets (approximately 80%), reflecting the robustness of the probabilistic data structures employed, such as Bloom filters and Count-Min Sketch.

A crucial observation is that with the defense mechanisms in place, the network was able to handle a higher total number of packets. This indicates that the system efficiently processes and discards malicious packets early in the pipeline, allowing legitimate traffic to be processed without significant delay. This efficiency is crucial in maintaining network performance and stability during an attack. Another important finding is the reduction in throughput consumed by attack traffic when the defense mechanisms were active. By quickly dropping attack packets, the system prevented network congestion and ensured that bandwidth and processing power were available for legitimate traffic. This is a critical factor in maintaining the quality of service and preventing potential service disruptions.

However, the study also revealed challenges associated with implementing and deploying these defense mechanisms. The complexity of programmable switches and the P4 language introduces a learning curve for network operators. Additionally, performance overheads and integration issues need to be carefully managed to ensure that the defense mechanisms do not negatively impact the overall network performance.

7. FUTURE WORK

Building on the findings of this study, several avenues for future research and development can be explored to enhance the effectiveness and deployment of P4-based defense mechanisms. Firstly, further optimization of the Bloom filter and Count-Min Sketch algorithms can be explored to reduce false positives and improve detection accuracy. Adaptive algorithms that can dynamically adjust parameters based on traffic patterns could be developed.

Extensive scalability testing is needed to evaluate the performance of the defense mechanisms in larger and more complex network environments. This includes testing with a higher number of hosts and more sophisticated attack patterns. Integrating the P4-based defense mechanisms with other network security systems, such as intrusion detection systems (IDS) and firewalls, could provide a comprehensive defense strategy. This integration would enable a more holistic approach to network security.

Developing user-friendly interfaces and management tools for programmable switches and P4 code could lower the barrier to entry for network operators. These tools could include visual dashboards for monitoring traffic patterns and configuring defense mechanisms. Conducting pilot deployments in real-world network environments would provide valuable insights into the practical challenges and benefits of implementing P4-based defense mechanisms. Collaboration with industry partners could facilitate these deployments and provide access to diverse network conditions.

Lastly, incorporating machine learning techniques to enhance traffic analysis and anomaly detection could improve the effectiveness of the defense mechanisms. Machine learning models could be trained to recognize evolving attack patterns and adapt the defense strategies accordingly. By addressing these areas in future work, the robustness, efficiency, and practicality of P4-based defense mechanisms against ICMPv6 attacks can be further enhanced, contributing to the overall security and stability of modern network infrastructures.

8. REFERENCES

[1] Döring, T., Stubbe, H., & Holzinger, K. (n.d.). *SmartNICs: Current Trends in Research and Industry*. Technical University of Munich. https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2021-05-1/NET-2021-05-1_05.pdf

[2] Kim, D., Lee, S., & Park, K. (2020). A case for SmartNIC-accelerated private communication. *KAIST*. https://doi.org/10.1145/3411029.3411034

[3] Khooi, X. Z., Csikor, L., Divakaran, D. M., & Kang, M. S. (2020). DIDA: Distributed In-Network Defense Architecture Against Amplified Reflection DDOS Attacks. *IEEE*. https://doi.org/10.1109/netsoft48620.2020.9165488

[4] Li, Y., Yang, W., Zhou, Z., Liu, Q., Zhao, L., & Li, S. (2022). P4-NSAF: defending IPv6 networks against ICMPv6 DoS and DDoS attacks with P4. *ICC 2022 - IEEE International Conference on Communications*. https://doi.org/10.1109/icc45855.2022.9839137

[5] Zhang, M., Li, G., Kong, X., Liu, C., Xu, M., Gu, G., & Wu, J. (2023). NETHCF: Filtering spoofed IP traffic with programmable switches. *IEEE Transactions on Dependable and Secure Computing/IEEE Transactions on Dependable and Secure Computing*, *20*(2), 1641–1655. https://doi.org/10.1109/tdsc.2022.3161015