World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(RESEARCH ARTICLE)

Check for updates

A modeling and simulation framework for the analysis of offensive and defensive strategies in computer networks

Moussa KOITA [1,*], Issa Traore [2], Oumar Y. MAIGA [3] and Mohamed Y. MAIZE [4]

[1] *University of Alassane Ouattara (UAO), Ivory Coast.*
[2] *University of Félix Houphouët-Boigny (UFHB), Institute of Mathematical Research.*
[3] *University of Sciences, Techniques and Technologies of Bamako (USTTB).*
[4] *Doctoral School of Science and Technology of Mali.*

World Journal of Advanced Research and Reviews, 2026, 31(02), 727–735

Publication history: Received on 29 June 2026; revised on 08 August 2026; accepted on 10 August 2026

Article DOI: https://doi.org/10.30574/wjarr.2026.31.2.2081

Abstract

This article proposes a simulation framework for a flexible evaluation of security strategies against denial-of-service (DoS) attacks in computer networks. The focus is on the SynFlood attack, a specific type of DoS that is one of the most common yet least understood threats in the field of IT security. The SynFlood attack exploits the connection-oriented functionality of the Transmission Control Protocol (TCP) to overwhelm the server with massive requests and prevent it from providing services to legitimate clients. The proposed framework aims to support network administrators in their efforts to design and evaluate the performance of new defense strategies against emerging SynFlood attack approaches. By combining the expressive power and friendliness of a visual modeling language with the robustness of a universal simulation language and its implemented infrastructure, such a framework favors communication between network experts and simulation experts, and allows them to focus on the specification of their strategies without the need to engage in cumbersome implementation efforts.

Keywords: Network security; Modeling and Simulation; SynFlood; HiLLS; DDOS attack.

1. Introduction

With the rise of the Internet of Things (IoT), smart connected systems (such as smart grids and smart cities), and cyber-physical systems (i.e., systems tightly integrating various physical and computing components via communication networks), security is becoming a major concern for their administrators, managers, and the billions of users and devices involved. One of the most common, yet still unresolved, security problems is the denial-of-service (DoS) attack [1], [2].

A denial-of-service (DoS) attack is a type of single-source attack that targets network infrastructure and prevents a server from responding to its clients. It involves sending millions of requests to a server from an invalid or spoofed IP address to slow it down [3]. It can target a computing resource, such as the CPU, or a network resource, such as the bandwidth of the victim's network link, or a combination of both. The impact of a DoS attack can range from a slight increase in service response time to total inaccessibility, sometimes with financial consequences for organizations heavily reliant on service availability.

A distributed denial-of-service (DDoS) attack is a distributed variant of the classic denial-of-service (DoS) attack. It involves using a set of geographically distributed compromised entities (often called bots, zombies, etc.) against one or more targets to cause a denial of service. All of these compromised machines, called bots or zombies, constitute a botnet.

* Corresponding author: Moussa KOITA.

In this type of DoS attack, the individual attack power of each compromised computer is aggregated and then used against a common target, thus amplifying the effect. Figure 1 illustrates the operation of a typical DDoS attack where the botnet is controlled by a master server.

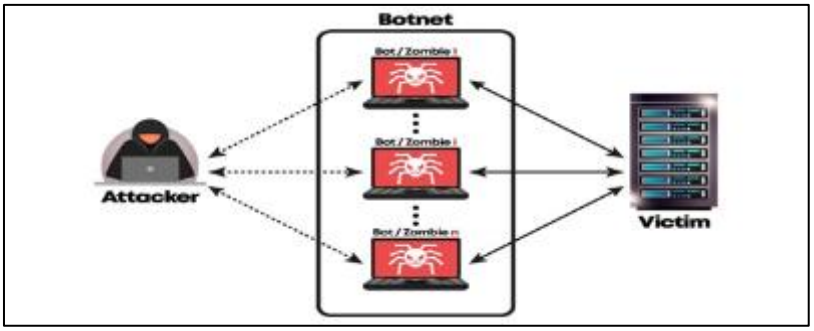

Figure 1 Typical structure of a botnet attack - Source: adapted from [4].

The Master takes control of an army of machines without the knowledge of their owners, usually by infecting them with a Trojan horse or backdoor program, and uses them simultaneously to attack a target server via the public Internet infrastructure.

Each bot will then launch a SYN flood attack, a technique that exploits a vulnerability in the Transmission Control Protocol (TCP) used by the Internet [5]. This vulnerability stems from TCP's "three-step handshake" principle, which, in the case of a normal connection, occurs in three phases (as illustrated in Figure 1.a):

- When initializing a TCP connection between a client and a server, the client sends a SYN message to the server, which is a vector (#seq, ACK flag, SYN flag) where #seq is chosen randomly by the client, the ACK flag = 0 and the SYN flag = 1.
- Next, the server allocates memory space for the transmission control block (TCB) and sends a SYN ACK message, a vector (#rseq, #ACK, ACK flag, SYN flag) where #rseq is chosen randomly by the server, #ACK = #seq + 1, ACK flag = 1 and SYN flag = 1
- Next, the client sends an ACK message, the vector (#ACK, #rseq + 1, 1, 0)

Establishing the connection ensures that both parties are ready to exchange data. A timer is triggered each time a SYN message is sent. If the SYN ACK response is delayed (longer than specified), the connection is dropped.

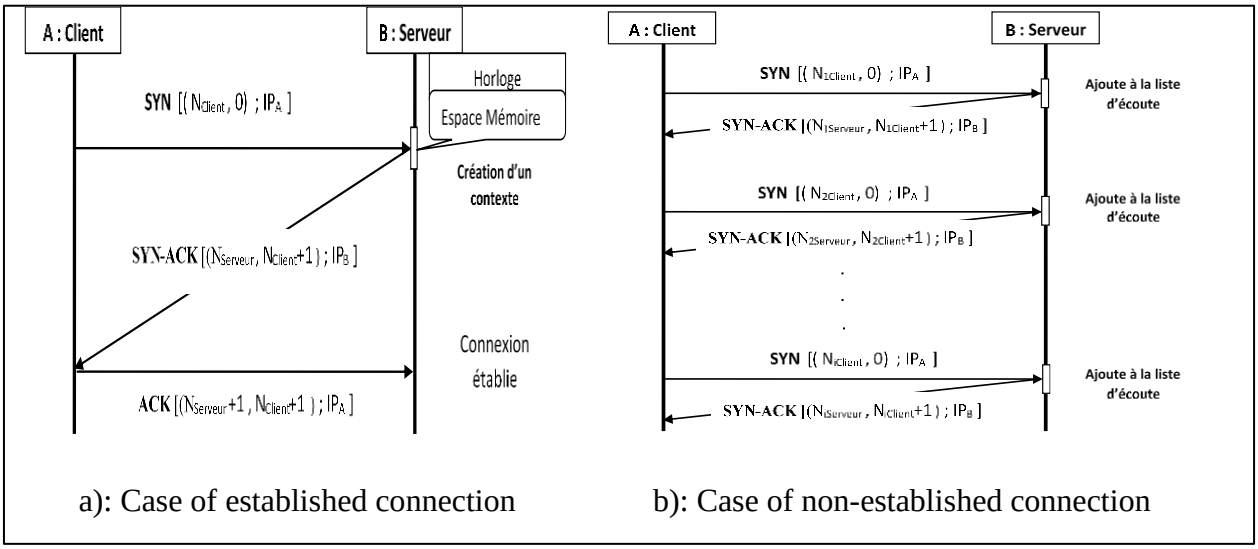

Figure 2 Principle of the handshake

A bot (acting as a malicious client) will ignore the last step and will not respond with an ACK message (Figure 2b). The server will then wait a certain amount of time before releasing the memory space allocated to the TCB (timeout). This strategy, while necessary to avoid the permanent allocation of unused memory space, also generates false positives. Indeed, the delay in receiving a SYN-ACK response can be due to network latency, even if the client is not malicious. The allocation by systems of a large number of resources for establishing multiple potential connections can lead to server failure.

Various strategies have been developed to counter this type of attack, and their performance depends on parameters such as network configuration, server capacity, attack frequency, etc. However, all these strategies are designed in an ad hoc manner, and it is recognized that there is a lack of methodological and operational framework for exploring performance-based security strategies [6]. This article proposes such a framework, allowing network experts and simulation experts to jointly explore the effectiveness of new defense strategies in the context of multiple attack configurations (including network structure and characteristics, as well as botnet structure and behavior). Therefore, this feature provides invaluable decision-making support for network administrators.

The rest of this article is organized as follows. Section 2 deals with related work. Section 3 presents the modeling formalism, namely the HiLLS language. Section 4 presents the basic metamodel and component models, as well as their operation. Finally, Section 5 describes the implementation and analysis of the results. The conclusion section summarizes the key contributions of this work, puts the results into perspective, and introduces future directions.

1.1. Related work

In [7], the approach consists of modeling the system using game theory, employing various strategies for the players and developing an intelligent defense that dynamically adjusts the system's parameters in the event of an attack. In [8], the approach consisted of modeling the strategy with a queue planning algorithm that differentiated attack requests from other types and allowed their rejection, thus avoiding the allocation of buffer space on these connections. In [9], the proposed detection scheme is based on SYN segment intensity measurements. The device periodically measures the number of SYN segments and compares it to a defined critical value.

There are many solutions against SynFlood attacks [10]. According to [11], these solutions are either server-based or router-based, and follow the following directions: (1) Optimized configuration thanks to improved system and router configuration. (2) Firewall architecture integrating a relay firewall and a semi-transparent gateway firewall. (3) Improved infrastructure. (4) connection establishment.

Server-side defense methods include: client pazzle [12], defensive programming [13], hop count filtering [14], path identifier [15], SYN caching [16] and [17], SYN cookies [18], detection and mitigation of SYN Flood [11] and [19], SynDefender [19] and other commercially available products [20], as well as SYN proxy [21].

For router-based methods, we have, among others: packet-distributed filtering [22], ingress filtering [23], referencing [24], and SAVE [25].

Some people have studied isolated attack strategies and defensive isolation strategies; our framework allows us to study both simultaneously and, moreover, to vary the network characteristics in order to search for the most effective strategy in different situations.

The main objective of this work is to establish a framework for evaluating network attack and defense strategies. The study will consider a synthesis of hacker behavior in the face of existing diversity, and effective defense strategies will be developed at the server level.

Table 1 Summary Table

Strategy / Location	Defense	Attack
Node-based	[17], [3]	[26], [27]
Network-based	[22], [23], [25]	[28]

Our unique approach lies in integrating attack and defense modeling with system simulation capabilities to quantify the potential impact of an attack and the performance of a defense. In this study, we propose establishing a generic framework for analyzing attack and defense strategies. This framework will reduce modeling and simulation efforts and facilitate collaboration between network and simulation experts.

Thus, when a network security expert proposes an algorithm to protect against denial-of-service attacks, this framework allows them to validate it by choosing the best defense strategy. It doesn't directly solve the denial-of-service problem itself, but rather addresses the lack of tools to study algorithm selection based on data types and specific network situations. In other words, it enables the selection of the most appropriate algorithm for each situation.

2. Modeling Language

This framework is based on a metamodel and a specification using a high-level language, making it usable with any discrete event simulation formalism.

HiLLS (High Level Language for System Specification) is a graphical formalism for specifying models of discrete event systems (DES) for analysis using methodologies such as simulation, formal analysis, and implementation [29]. Designed to simulate a wider range of DESs with greater expressiveness and communicability, HiLLS uses concrete graphical notations from software engineering, in accordance with DEVS concepts, to make it accessible to a diverse user community [29].

HiLLS is a graphical language whose abstract syntax is derived from the integration of concepts from systems theory and software engineering [29]. As illustrated in Figure 3, HiLLS offers a unified abstract syntax for describing SEDs (Semantic Event Systems), a concrete graphical and textual syntax, and three major semantic domains: the Unified Modeling Language (UML), the specification of discrete event systems (DEVS), and formal analysis [29].

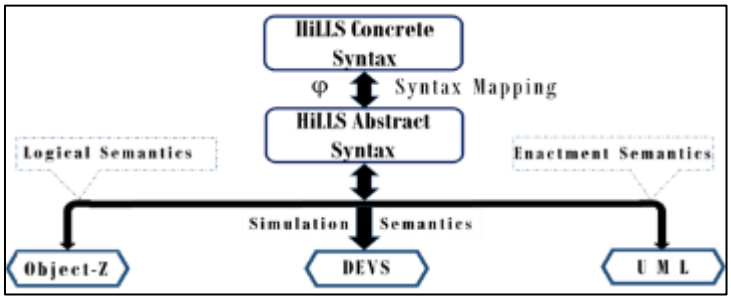

Figure 3 Overall view of the HiLLS Formalism – Source: adapted from [29].

3. Basic Model

The framework described above is designed for modeling and simulating the behaviors and recognition strategies of SYN-Flood attacks.

Detecting Synflood attacks is generally complex. Attack strategies and their mechanisms can vary greatly. Therefore, it is difficult to use a uniform model to describe different attacks. To identify common characteristics across different strategies, a conceptual framework is used to describe some of them.

The HiLLS metamodel (Figure 4) presents the hierarchical composition of the components of our SynFlood attack model as well as their input/output interfaces. The SynFlood attack model comprises five components and different types of communication between the entities.

The set of components is defined as follows: Component ∈ {Network, Server, normal, hacker, Message}. The Network, Server, Normal, and Pirate elements are of type HSystem, while Message is of type HClass. Furthermore, Server and Client are modeled as system nodes, with the Client class being divided into two categories: Legitimate Client and Rogue Client.

The Network entity, modeled as an Hsystem, is characterized by two complex attributes: packet and latency. The packet attribute corresponds to a message of type HClass, handled by the system's nodes. The latency attribute, on the other

hand, is modeled as a probability distribution of type HClass, representing the network's temporal behavior. This distribution is configurable and can be defined using one or more parameters.

The Server entity has a complex Buffer attribute which is a queue, an instance of the HClass Memories.

Note that the HClass Memories is composed of another HClass Cell (Cell) whose two important attributes are the node identifier and the timer (Timer).

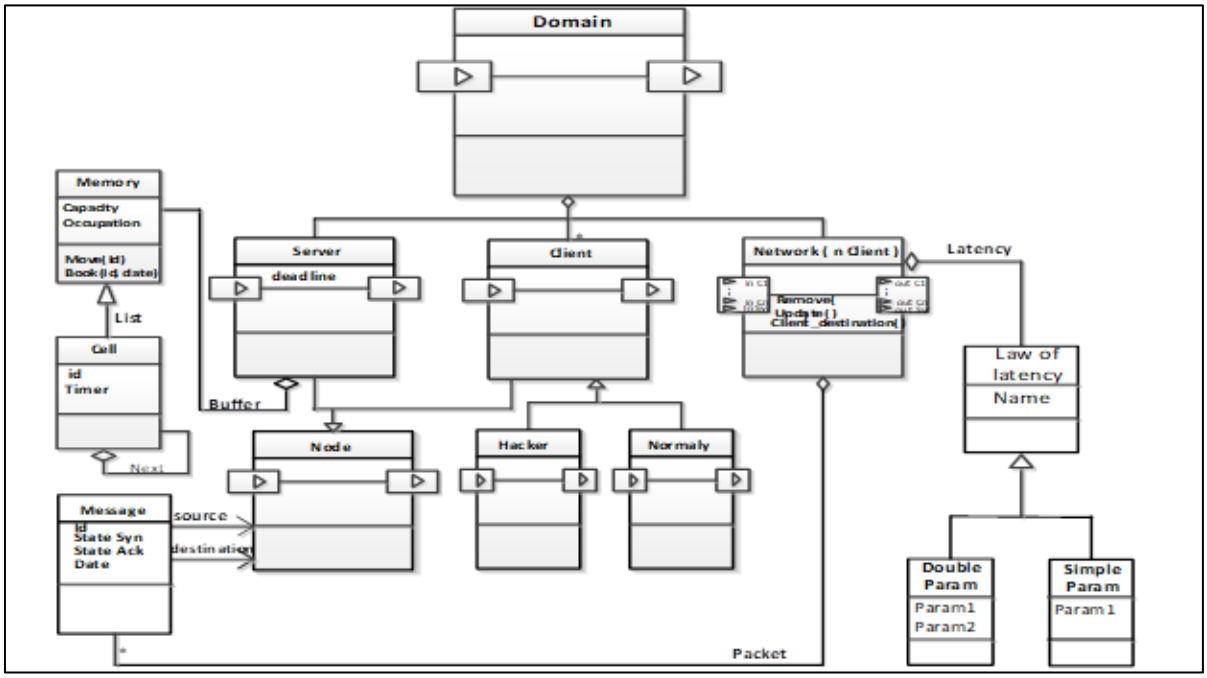

Figure 4 The HiLLS metamodel of the system

4. Application

This article presents a framework called " SynFlood attack Framework" for testing the validity of SYN flood defense algorithms and comparing the effectiveness of two algorithms in different situations. As mentioned in the introduction, this framework will allow network experts, even without simulation experience, to simulate their algorithms and evaluate their validity. Once the design of our SynFlood attack Framework is finalized, we will be able to test the algorithm of [7].

4.1. Multi-scenario case study

The algorithm described in this article, which proposes a game between legitimate users and attackers, is structured around four strategies (see Table 2: Game Strategies). In this game, legitimate players cooperate to avoid saturating the server buffer and thus optimize network efficiency. Requests, whether legitimate or attack-related, are injected into the system. If the buffer has free space, new semi-open connections are placed there. Conversely, if the buffer is full, incoming requests are blocked. The system's behavior is simulated according to four strategies, for different values of m (buffer length: maximum number of semi-open connections), h (duration of semi-open connections in seconds) (Table 2), and other parameters.

Table 2 Illustration of the strategy to be tested: Source [7].

Game strategies		
	Number of semi-open connections (m)	Maintenance time (h)
Strategy 1	Increase	Increase
Strategy 2	Increase	Decrease

Strategy 3	Decrease	Increase
Strategy 4	Decrease	Decrease

By considering the following metrics [7]: The probability of loss (P_{loss}), the percentage (P_r) occupancy of the regular request buffer and the percentage (P_a) occupation of the attack requests buffer. With:

$$P_{loss} = \frac{\text{The total number of blocked packets}}{\text{The total number of packets entered in the server}}$$

The average ratio between the number of semi-open connections created by regular queries and the total number of queries in the server is called normal demand buffer occupancy percentage (P_r).

$$P_r = \frac{\text{The number of semi} - \text{open connections created by regular queries}}{\text{The total number of requests in the server}}$$

The average ratio between the number of half-open connections created by the attack requests and the total requests in the server is called the percentage of the attack request buffer occupancy (P_a).

$$P_a = \frac{\text{The number of semi} - \text{open connections created by attack requests}}{\text{The total number of requests in the server}}$$

The objectives of the work:

(1) Reduce the value of the blocking of the request. (2) Increased percentage and buffer occupancy time by regular requests. (3) Reduced percentage and buffer occupancy time by attack requests. (4) Maximizing the value of the function «Goal»:

$$F(t) = \frac{P_r}{(P_{loss} * P_a)}$$

In the defense mechanism always described by [7]; the server first using the initial parameters m and h starts the service. The strategies are as in the table. A weight W[i] with i = 1, 2, 3 and 4 is assigned to each movement.

The value of Pr, Pa and Ploss is estimated at each instant t and consequently that of the function «goal» F (t). If the new goal value is improved relative to the value of the objective function in the previous period, the weight value of the selected strategy in the last period increases and vice versa. For the next period, the best strategy is selected based on the best values in the W.

Initially for the server we will retain the following values which according to [9] are considered as the default values. Timer (SYN_RECEIVED timeout) = 120, the buffer (Backlog queue length) = 128.

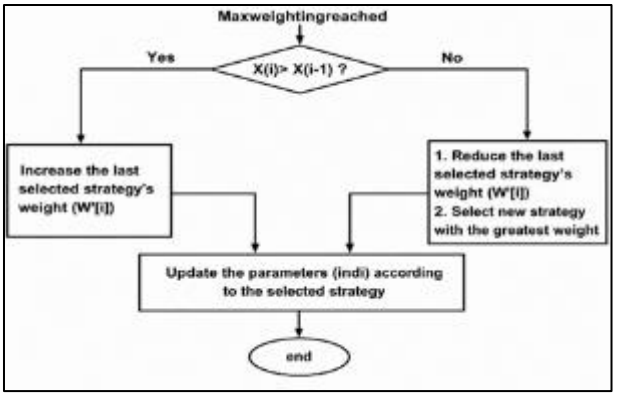

Figure 5 Defense Algorithm [7]

4.2. Simulation Results

The simulation results below were obtained under the following conditions:

- 5,000 nodes distributed according to the following scenarios
- Scenario 1: 2,000 hackers and 2,500 legitimate users
- Scenario 2: 3,000 hackers and 2,000 legitimate users
- Scenario 3: 2,000 hackers and 3,000 legitimate users
- The simulation duration is 50 s.
- The number of SYN packets sent per source address per unit of time (s) is 2. This value can vary depending on whether it is an attack or not. Attackers often increase it considerably to circumvent defense techniques.
- The interval between transmissions is random. This parameter is chosen for the same reasons as the number of packets.
- The duration of the SYN-RECEIVED state for each request has been set to 75 seconds, which corresponds to the default duration on some operating systems.
- The queue (backlog) length is 1024.

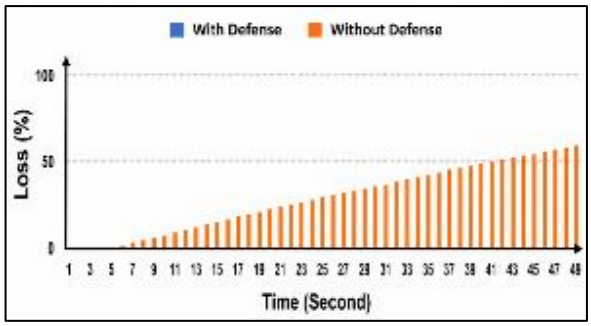

Figure 6 Request loss rate

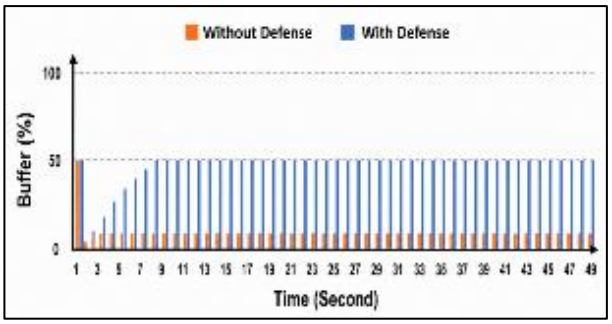

Figure 7 Server buffer usage with a normal request

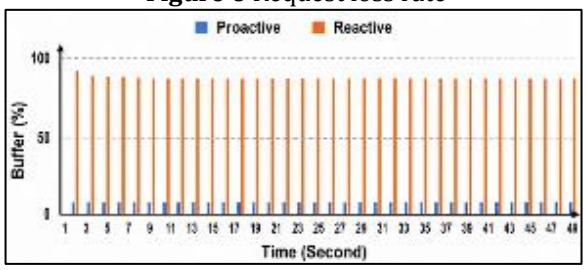

Figure 8 Server buffer occupancy rate (without protection)

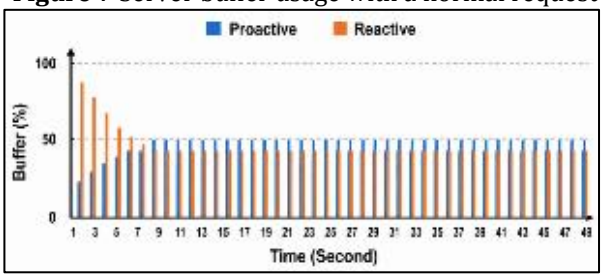

Figure 9 Server buffer occupancy rate (with protection)

In this first scenario, focused on the evolution of the percentage of lost packets, we observe that this percentage is zero throughout the simulation when the defense strategy is active. This explains why the server consistently receives the SYN RECEIVED signal for all requests. Indeed, the algorithm has time to evaluate the state of the memory. Conversely, in the absence of defense, we observe an increase in the number of lost packets.

Figure 7, which focuses on client activity, shows that its memory space usage is higher when the defense strategy is activated and remains consistently higher than that observed otherwise.

Figures 8 and 9 illustrate server memory usage. When the defense is disabled, the attacker's memory usage remains significantly higher than that of the legitimate client. This is because the attacker never adheres to the "three-step handshake" principle; that is, they never receive the final acknowledgment (ACK), thus preserving the SYN state received by the server for a longer period. When the defense is enabled, the attacker's memory usage is high during the first few seconds, while the legitimate client's activity balances with its defense strategy.

5. Conclusion

This study highlights the complexity and persistence of denial-of-service attacks, particularly SynFlood attack, whose mechanisms exploit fundamental vulnerabilities in network protocols. Despite the diversity of existing solutions, their effectiveness remains highly dependent on the execution context, which limits their universal applicability.

To address this limitation, we proposed a modeling and simulation framework based on the HiLLS language, enabling the joint analysis of attack and defense strategies in a controlled and configurable environment. This framework promotes a better understanding of the interactions between attackers and defenders, while reducing development effort through a unified approach accessible to both network and simulation experts.

The results obtained show that integrating adaptive strategies significantly improves system performance, particularly by reducing packet loss and optimizing buffer usage by legitimate requests.

Thus, this work does not claim to definitively solve the problem of DoS attacks, but rather offers a relevant methodological tool for evaluating and selecting defense strategies adapted to various contexts. It therefore constitutes a valuable decision-making aid for network administrators and opens up interesting perspectives for integrating intelligent techniques, particularly those based on machine learning, into the dynamic management of cybersecurity.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] David, J., & Thomas, C. (2020). Detection of distributed denial of service attacks based on information theoretic approach in time series models. Journal of Information Security and Applications, 55, 102621.

[2] Al-zubidi, A. F., Farhan, A. K., & Towfek, S. M. (2024). Predicting DoS and DDoS attacks in network security scenarios using a hybrid deep learning model. Journal of Intelligent Systems, 33(1), 20230195.

[3] Elleithy, K. M., Blagovic, D., Cheng, W. K., & Sideleau, P. (2005). Denial of service attack techniques: Analysis, implementation and comparison.

[4] Emmanuel, C. O., Nikos, V., Ogu, C., & Ajose-Ismail, B. M. (2020). On the internal workings of botnets: A review. International Journal of Computer Applications, 138(4), 39-43.

[5] Kumar, M., Panwar, A., & Jain, A. (2012). An analysis of TCP SYN flooding attack and defense mechanism. International Journal of Engineering Research & Technology.

[6] Mirkovic, J., Hussain, A., Wilson, B., Fahmy, S., Reiher, P., Thomas, R., ... & Schwab, S. (2007, June). Towards user-centric metrics for denial-of-service measurement. In Proceedings of the 2007 workshop on Experimental computer science (p. 8). ACM.

[7] Abbasvand, S., Hashemi, S. N. S., & Jamali, S. (2014). Defense against SYN-flooding attacks by using game theory. Indian Journal of Science and Technology, 7(10).

[8] Jamali, S., & Shaker, G. (2014). Defense against SYN flooding attacks: A scheduling approach.

[9] Ramkumar, B. N., & Subbulakshmi, T. (2021). TCP SYN flood attack detection and prevention system using adaptive thresholding method. In ITM Web of Conferences (Vol. 37, p. 01016). EDP Sciences.

[10] Sing, C., & Jain, A. K. (2024). A comprehensive survey on DDoS attacks detection & mitigation in SDN-IoT network. e-Prime-Advances in Electrical Engineering, Electronics and Energy, 8, 100543.

[11] Yang, C. H., Wu, J. P., Lee, F. Y., Lin, T. Y., & Tsai, M. H. (2023). Detection and mitigation of SYN flooding attacks through SYN/ACK packets and Black/White lists. Sensors, 23(8), 3817.

[12] Juels, A., & Brainard, J. G. (1999, March). Client puzzles: A Cryptographic countermeasure against connection depletion attacks. In NDSS (Vol. 99, pp. 151-165).

[13] Qie, X., Pang, R., & Peterson, L. (2002). Defensive programming: Using an annotation toolkit to build DoS-resistant software. ACM SIGOPS Operating Systems Review, 36(SI)

[14] Jin, C., Wang, H., & Shin, K. G. (2003, October). Hop-count filtering: an effective defense against spoofed DDoS traffic. In Proceedings of the 10th ACM conference on Computer and communications security (pp. 30-41). ACM.

[15] Yaar, A., Perrig, A., & Song, D. (2003, May). Pi: A path identification mechanism to defend against DDoS attacks. In Security and Privacy, 2003. Proceedings. 2003 Symposium on (pp. 93-107). IEEE.

[16] Lemon, J. (2002, February). Resisting SYN Flood DoS Attacks with a SYN Cache. In BSDCon (Vol. 2002, pp. 89-97).

[17] Criscuolo, P. J. (2000). Distributed denial of service, tribe flood network 2000, and stacheldraht CIAC-2319, Department of Energy Computer Incident Advisory Capability (CIAC). UCRL-ID-136939, Rev. 1., Lawrence Livermore National Laboratory.

[18] Divakaran, D. M., Murthy, H. A., & Gonsalves, T. A. (2006, September). Detection of SYN flooding attacks using linear prediction analysis. In Networks, 2006. ICon'06. 14th IEEE International Conference on (Vol. 1, pp. 1-6). IEEE

[19] Check Point Software Technologies Ltd; http://www.checkpoint.com/products/firewall-1 Netscreen 100 Firewall.

[20] Netscreen 100 Firewall Appliance, http:// www.netscreen.com/ accessed: June 15, 2026.

[21] Take proactive steps to prevent DDoS attacks, Feb.6,2004, [online]. http://www.computerworld. com/s/article/89932/Mydoom lesson Take proactive steps to prevent DoSattacks? taxonomyId =017.

[22] Mishra, P., Pilli, E. S., Varadharajan, V., & Tupakula, U. (2023). Distributed DDoS Detection and Mitigation Techniques in Software Defined Networks. IEEE Access, 11, 48430-48450.

[23] Senie, D., & Ferguson, P. (1998). Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. Network.

[24] Ioannidis, J., & Bellovin, S. M. (2002, February). Implementing Pushback: Router-Based Defense Against DDoS Attacks. In NDSS.

[25] Li, J., Mirkovic, J., Wang, M., Reiher, P., & Zhang, L. (2002, June). SAVE: Source address validity enforcement protocol. In INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE

[26] NESG - Network Engineering & Security Group (2013): NETA: A NETwork Attacks Framework. Architecture and Usage. University of Granada.

[27] Pei, L., Li, C., Hou, R., Zhang, Y., & Ou, H. (2013). Computer simulation of denial-of-service attack in military information network using opnet. In Proceedings of the 3rd International Conference on Multimedia Technology (ICMT'13).

[28] Wang, H., Zhang, D., & Shin, K. G. (2002, June). Detecting SYN flooding attacks. In INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE (Vol. 3, pp. 1530-1539). IEEE.

[29] Aliyu, H. O., Maïga, O., & Traoré, M. K. (2016). The high-level language for system specification: A model-driven approach to systems engineering. International Journal of Modeling, Simulation, and Scientific Computing, 7(01), 1641003.