Securing Software-Defined Tactical Networks: A Cyber Defense System

Sean Kloth*†, Paulo H. L. Rettore†, Philipp Zißner†, Bruno P. Santos‡, and Peter Sevenich†
*Institute of Computer Science 4, University of Bonn, Bonn, Germany
†Department of Communication Systems, Fraunhofer FKIE, Bonn, Germany
‡Department of Computer Science, Federal University of Bahia, Salvador, Brazil
Email: s6seklot@uni-bonn.de, {sean.kloth, paulo.lopes.rettore, philipp.zissner, peter.sevenich}@fkie.fraunhofer.de, and bruno.ps@ufba.br

***Abstract*—The deployment of Software-defined Networking (SDN) in Tactical Networks (TNs) enables communication in adverse and heterogeneous scenarios. However, SDN is vulnerable to cyber threats, particularly at the edge. This article addresses the critical issue of mitigating cyber attacks in networked environments by proposing a comprehensive framework comprising three distinct agents: Cyber Attack Agent (CAA), Cyber Defense Agent (CDA), and Network Manipulation Agent (NMA). The CAA simulates sophisticated attacks, including network reconnaissance, flow table flooding, and Distributed Denial-of-Service (DDOS) attacks, with the capability of evaluating the success of their attack. The CDA leverages either a threshold-based mechanism or a machine learning model, such as Long Short-Term Memory (LSTM), for robust anomaly detection and response mechanisms. The NMA enhances the capabilities of the CDA by stimulating network conditions in TN. The proposed framework is evaluated across multiple network topologies, and metrics such as anomaly detection accuracy, attack efficiency, and system adaptability are analyzed by comparing the threshold-based and learning-based mechanisms.**

***Index Terms*—Tactical networks, Software-defined Networking, Resilience, Cyber security**

I. Introduction

Software-defined Networking (SDN) enables direct programmability by decoupling the control and data planes, providing flexibility across various contexts, including civilian and military applications [1]–[3]. This independence from specific technologies and vendors facilitates the integration of diverse communication methods. However, these methods often demand specialized SDN implementations to address their unique requirements [4].

Several factors, including mobility, radio capabilities, terrain, security risks, and signal interference, are influenced by fluctuations in Tactical Networkss (TNs). As a result, tactical edge systems must remain robust against network disruptions, network variability, cyber threats, and dynamic policies. The integration of SDN into Tactical Networks (TNs) leads to Software-defined Tactical Network (SDTN), introducing further challenges such as network vulnerabilities, Single Point of Failure (SPoF), Controller Placement Problem (CPP), shifting topologies, and increased management overhead. A critical vulnerability in SDTNs arises when controllers are targeted by malicious actors, potentially disrupting or manipulating communications. To address this, resilient controllers must detect attacks and implement countermeasures effectively.

This work is preceded by the related studies [5]–[9] that highlight the importance of improving SDTN resilience by incorporating tactical networks characteristics that are often overlooked in SDN-focused studies. Therefore, we propose a defense strategy that quickly identifies and mitigates abnormal control plane activity by activating a backup controller or disabling compromised switch ports. The framework incorporates three distinct agents: Cyber Attack Agent (CAA), Cyber Defense Agent (CDA), and Network Manipulation Agent (NMA), each designed to enhance the network's ability to detect, respond to, and mitigate cyber threats. Our contributions are summarized as follows:

- Utilization of three agents: 1) The CDA to monitor the network and analyze incoming traffic to detect ongoing cyber threats, including network reconnaissance, flow table flooding, and Distributed Denial-of-Service (DDOS) attacks. It employs either a threshold-based or a learning-based mechanism to safeguard the SDN controller. 2) The CAA is designed to perform network reconnaissance and, based on the collected information, execute either a flow table flooding attack or a DDOS attack to challenge the CDA. 3) The NMA facilitates the modification of the network topology and the links between hosts and switches.
- A methodology for a range of defensive responses, such as modifying the flow tables of switches, isolating compromised switches, or rebuilding the network using a backup controller to ensure continued network functionality.
- A comparison between the threshold-based and the learning-based solutions.
- Experimental results over different network conditions quantifying the effectiveness of both the CDA and CAA proposed.

The paper is organized as follows: Section II reviews relevant literature. Section III describes the agents' design. Section IV presents the findings, and Section V concludes with a summary and future perspectives.

ISBN 978-3-903176-72-0

II. Related Work

Cyber defense in SDTN has been an active area of research. In particular, Kloth et al. [8] introduced a cyber defense approach leveraging a CAA and a CDA to enhance controller resiliency in SDTNs. The CAA is responsible for attack preparation by generating attack packets from a host or switch, focusing on DDOS-based assaults. Their evaluation concluded that data plane attacks are more effective due to their ease of execution and amplification potential. The CDA, on the other hand, detects and responds to DDOS attacks by analyzing key network traffic metrics such as entropy, packet-in request rates, response times, and packet counts per switch. Using a threshold-based model, the CDA can identify anomalies and take action, such as isolating compromised switches or reassigning them to backup controllers. Their evaluation demonstrated the effectiveness of these metrics in detecting ongoing DDOS attacks. However, threshold-based systems require precise tuning and are not adaptable to dynamic network conditions such as topology changes, throughput variations, and latency fluctuations.

Pérez-Díaz et al. [10] proposed an SDN-based security architecture to detect and mitigate low-rate DDOS (LR-DDoS) attacks using Machine Learning (ML). LR-DDoS attacks, which exploit network resource allocation vulnerabilities through periodic malicious traffic bursts, are hard to detect with traditional methods. The authors developed a flexible IDS/IPS within an SDN environment, employing six ML models. Trained on the Canadian Institute for Cybersecurity Denial-of-Service (DoS) dataset, the system achieved a 95% detection rate. Implemented on an Open Network Operating System controller within a Mininet simulation, the study demonstrates the effectiveness of ML for dynamic attack detection and mitigation in SDNs.

Li et al. [11] proposed a two-level CNN-LSTM model to enhance DDOS detection. The Convolutional Neural Network (CNN) extracts spatial features, identifying local anomalies in packet distributions, while the Long Short-Term Memory (LSTM) models temporal dependencies to detect attack trends. This hybrid deep learning approach improves on traditional ML methods by automatically extracting high-level attack signatures, reducing reliance on manual feature selection. The study shows superior attack detection performance compared to conventional techniques. Tang et al. [12] present FTODefender, a system designed to detect and mitigate Low-rate Flow Table Overflow (LFTO) attacks in SDN. LFTO attacks exploit the limited capacity of SDN flow tables by sending small packets that cause overflow. FTODefender consists of two components: the Detector, which identifies LFTO attacks using the CRITIC method to analyze flow table features, and the Mitigator, which uses LightGBM-LR to classify and remove malicious flow rules and block attacker IPs. Tested in a Mininet environment, FTODefender detected attacks with 97% accuracy and reduced malicious flow proportions with minimal computational overhead.

A common threat to controllers is the DoS attack. Abdullah et al. [13] evaluate how controllers, including Opendaylight, POX, and RYU, perform under DoS attacks. Using metrics such as latency, throughput, and bandwidth, the study analyzes the impact of attacks but does not propose mitigation strategies. Sahoo et al. [14] evaluates seven ML techniques, concluding that Learning Rate (LR) provides the highest precision, while Random Forest (RF) performs faster but less accurately. Feature-based approaches have also been proposed. Yue et al. [15] developed DoS detection using flow table features such as entropy and flow similarity, tested with various ML models.

The paper [16], [17] models tactical network dynamics using Markov-based state machines to represent both message flows and network conditions. Model A defines message types as states, where transitions occur probabilistically based on Quality-of-Service (QoS) priorities, ensuring differentiation between routine and critical messages. Model B models network states, with transitions between different data rates and disconnections reflecting real-world variations. Model A | B integrates both, using Markov chains to dynamically adapt message transmissions based on evolving network conditions. This approach enables efficient queuing and traffic management, ensuring prioritized message delivery under fluctuating connectivity.

In this paper, we are tackling the challenge of tactical networks with constrained links by adding the NMA. The CAA is also enhanced, including a network reconnaissance with machine learning algorithms detecting the type of topology to improve the attacker's knowledge of the target network. Finally, the CDA implements machine learning algorithms in order to detect the network anomalies over changes in link and topology conditions.

III. Design of the Agents

Resilience in SDTNs refers to a controller's ability to withstand attacks and provide a suitable response. Our strategy for addressing resilience comprises the design of a CDA capable of monitoring, analyzing characteristics, detecting anomalies, and reacting. In addition, a CAA is proposed to create a successful attack challenging CDA, while NMA is able to vary the network topology and link quality.

A. Cyber Attack Agent (CAA)

The creation of cyber threats is the main task of the CAA, challenging the controller of the SDTN. We introduce different attack vectors, including DDOS attacks, network reconnaissance, and flow table flooding. The CAA implements four main steps described in Fig. 1: Network reconnaissance, attack preparation, attack, and evaluation. Next, we describe the network reconnaissance, the attack preparation, the attack composed by a flow table flooding, a DDOS attack, and finally, the attack evaluation.

1) Network Reconnaissance and Evaluation: Network reconnaissance, gathering information to plan future operations as shown in the MITRE ATT&CK reconnaissance, forms the basis of the CAA. Considering a scenario with a group

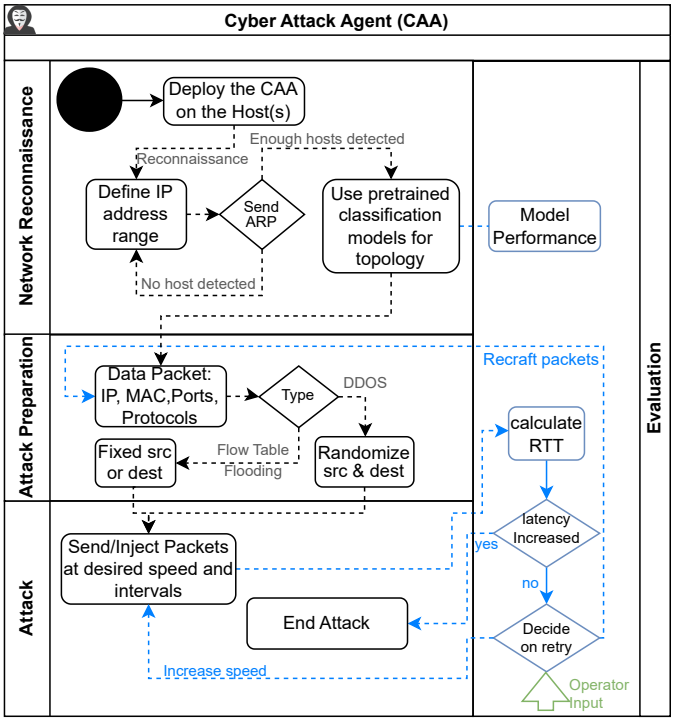


Fig. 1: Design of the Cyber Attack Agent (CAA).

of vehicles, we deploy an active adversary in the network reconnaissance. Initially, the attacker tries to get information about the company (or network topology) to estimate the company's size and identify potential targets together with the node's location. This can severely compromise the security of the convoy. The mapping of the network structure can reveal the principles of the networks.

The CAA uses Nmap, a tool for scanning network entities that is practical and simple. Address Resolution Protocol (ARP)-requests are generated, given a range of IP addresses to find all hosts within the network. The extracted information includes the MAC address, Round Trip Time (RTT), and IP addresses. To map the network topology, the CAA utilizes the RTTs. With the topology, we can improve the effectiveness of the attack. For example, in disconnected networks, DDOS attacks are less effective as the amplification effect is missing. Therefore, a flow table flooding attack might be more effective.

The more information the attacker has about the network, the easier it is to avoid detection, especially in larger networks, as it is less difficult to hide malicious activities among normal traffic. To this end, the CAA deploys a classification model using the normalized RTT. To avoid misleading data (outliers) due to the simulation environment, we preprocess the data by replacing all RTT values lower than the RTT for neighboring hosts with interpolated data. The resulting models are an abstraction of the underlying network topology given the position (in terms of connection, RTT) of the CAA, allowing the attacker to compare them to new reconnoitered data.

2) Preparation: Based on the information gained from the network reconnaissance, the CAA can improve the attack performance. Using this information allows the CAA to identify targets and IP addresses that force flow table misses. With this, the two attacks can be prepared as follows.

a) DDoS attack: The goal of a DDOS attack is to exhaust the controller's memory, CPU, or bandwidth. To this end, the CAA floods the controller with *packet-in* requests. In SDN, the network is separated into the control plane, which defines how the packets are forwarded, and the data plane that forwards the packets. The first plane enables communication between hosts through flow tables installed in switches. The installed entries include information about the routing path. For all new packets that cause a flow table miss, the corresponding switch sends a *packet-in* request to the controller to request the matching flow table entry. This process is part of the control plane. The CAA can induce flow table misses by crafting packets at the data plane, as shown in [8], ensuring effectiveness while minimizing assumptions. The attack's impact amplifies across connected switches, disrupting normal traffic by exhausting bandwidth. Since CAA only requires network access, switch security flaws are irrelevant. Given SDN architecture, data plane access is easier than control plane compromise. In SDTN, controller exhaustion may be limited by dynamic conditions, but delayed detection enhances the attack's impact.

b) Flow table flooding: The agent can craft packets that enable a flow table flooding attack based on a successful network reconnaissance. A flow table entry is installed when a route between the source and destination exists. Thus, we can craft packets with a fixed destination IP address while varying the source IP address, MAC addresses, protocol, and ports. Similarly, the agent can iterate over all discovered hosts within the network to give the attacker a broader attack surface.

3) Attack and Evaluation: After the preparation phase, the chosen attack is executed and assessed. To ensure CAA effectiveness, attacks must be evaluated based on their impact on the network. Since the CAA operates in a TN, the attacker's proximity enables an estimation of connected hosts. By identifying their IP addresses, the attacker defines a search range, adjusting it until reconnaissance succeeds.

The attacks aim to exhaust the controller's memory, CPU, or bandwidth. Bandwidth depletion is assessed by monitoring the RTT of an existing connection, potentially caused by flow table flooding. The CAA can reuse an installed entry to track RTT. Controller resource exhaustion is tested by measuring the time required to install new flow table entries via a secondary host connected to a different switch. Attack success is determined by an increased RTT. If no change occurs, manual adjustment is required, as failure causes remain uncertain. The operator can modify packet characteristics to induce more flow table misses or increase transmission speed to stress the controller or bandwidth more effectively.

B. Cyber Defense Agent (CDA)

The proposed CDA, described in Fig. 2, monitors the traffic in a SDTN, trying to detect malicious activities and responding appropriately to them. Notice that the controller cannot access the traffic information from the user network. Thus, we focus on information that can be extracted from the controller. The

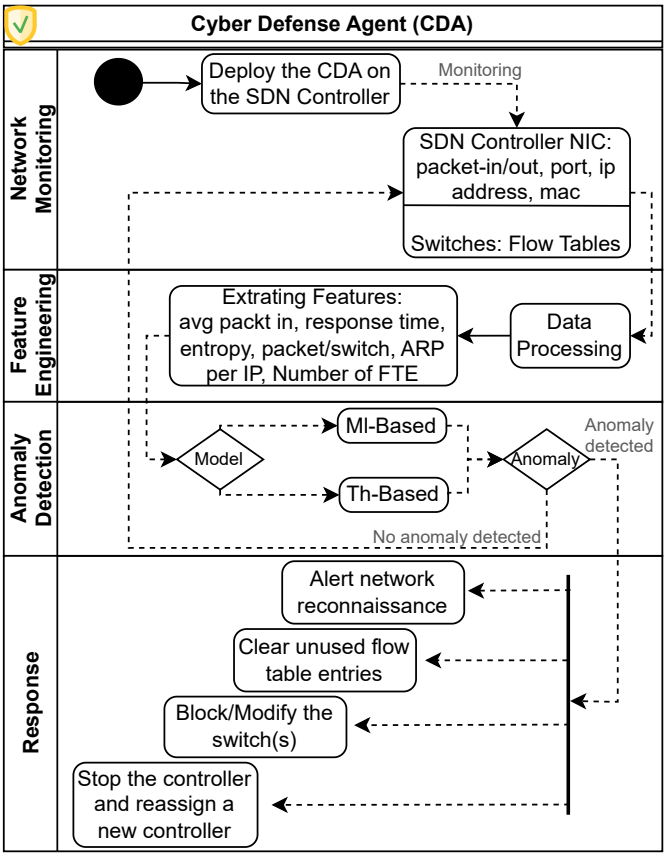


Fig. 2: Design of the Cyber Defense Agent (CDA).

CDA consists of four parts: *Network Monitoring*, *Feature Engineering*, *Anomaly Detection Mechanisms*, and *Response Mechanisms* as described below. The CDA enhancements include the capability to detect flood attacks on the flow table and the use of a machine learning-based model to detect DDOS or flood attacks on the flow table.

1) Network Monitoring: After deploying the CDA on the controller, the monitoring phase collects all relevant information from the IP traffic using a packet sniffer designed to respond instantly to received packets. The packet sniffer uses a Python extension module, PCAPy, to get the IP traffic. We use PCAPy over other well-known libraries for packet sniffing, such as Pyshark or Scapy, as we found out it provides the fastest processing of incoming packets after evaluating all tools. Due to the number of packets transmitted during such an attack, we need fast processing to avoid buffers or other delays in data processing.

2) Feature Engineering: In sequence, the feature engineering processes and temporarily store the logs to extract the following metrics *Entropy of packet-in requests*, *Average number of packet-in requests*, *Average response time for packet-in requests*, *Identification of compromised switches*. The authors explored these metrics in [8] and thus, due to space limits, this paper will focus on the additional functionalities designed, such as *Detection of network reconnaissance*, *Testing known SDN-DDoS features in SDTN*, and the *Flow table flooding detection*. All proposed features can be used either in a threshold-based or learning-based system.

a) Detection of network reconnaissance: Typically, network reconnaissance is done by sending ARP requests to different IP addresses. Using this information, we can keep track of the number of ARP requests per IP address. Additionally, the CAA measures the number of unique destinations, monitors the flow table entries for the IP addresses, and checks whether the entries are used or installed. An IP address with many ARP requests without the appropriate number of flow table entries can indicate an ongoing network reconnaissance. This is formally described in (1).

$$N_{\text{ARP}}(i) \gg N_{\text{ARP}}(j), \quad \forall j \neq i \tag{1}$$

Where:

- $N_{\text{ARP}}(i)$: The number of ARP requests sent by IP address i.
- i, j are switches in the network.

b) Flow table statistics: A flow table flooding attack is executed by varying the source or destination while keeping the counterpart at the same address. Intuitively, the flooding process can be detected by identifying a drastic increase in flow table entries. A fixed source IP address can reveal the attacker's location, while a fixed destination IP address can help identify a host that is not compromised, thereby reducing false positives. This is described in (2):

$$\begin{aligned} &\Delta N^t_{\text{Entries}}(i) = \\ &\quad N^t_{\text{Entries}}(i) - N^{t-1}_{\text{Entries}}(i) \gg \Delta N^t_{\text{Entries}}(j), \quad \forall j \neq i \end{aligned} \tag{2}$$

Where:

- $N^t_{\text{Entries}}(i)$: The number of flow table entries for switch i at time t.
- i, j are switches in the network.

3) Anomaly detection mechanism: We utilize the previously defined features to support the detection of anomalies in the network. The design of the features is intended to indicate an ongoing DDOS attack, a flow table flooding attack, or a network reconnaissance. If the features indicate an attack, a response is triggered. The detection mechanism is either threshold- or learning-based. For the threshold-based system, an attack is detected if the features exceed a pre-defined threshold (A_{TH}). The monitoring results of the switches are evaluated to identify compromised switches.

The CDA extends this naive solution by implementing the capability to learn the traffic pattern and classify normal and abnormal traffic usage. As the network features are calculated per switch, the CDA can directly identify the compromised switch. Thus, each data sample consists of the source switch and the corresponding features. To ensure a good model performance, the CDA deploys individual models for detecting a DDOS attack and detecting a flow table flooding attack, respectively. With this separation, the CDA can guarantee that the different attack features do not interfere with each other. The LSTM was chosen in the learning-based approach as the data is collected as a time series, and the agent must

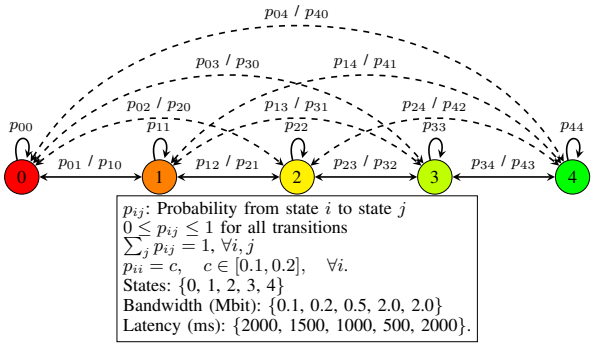


Fig. 3: State transition diagram with probabilities p_{ij}.

classify it. LSTMs are a specialized type of Recurrent Neural Networks (RNNs) with the capability of learning long-term dependencies. They utilize memory cells with gates to control the flow of information, enabling the classification of time series data.

4) The response mechanism: The final step involves implementing an appropriate response to the attack, which may include actions such as blocking compromised ports, isolating affected switches, clearing flow tables from switches, or reassigning uncompromised switches to a backup controller. This process is executed continuously to maintain resilience. If the anomaly detection flags an ongoing attack, a response must be triggered appropriately for the specific attack. These include stopping the switch from accepting new packets, clearing unused flow table entries, or removing the switch from the network. If the switch-specific responses fail, uncompromised switches can be reassigned to a backup controller. If network reconnaissance is detected, the attacker needs to be removed from the network.

C. Network Manipulation Agent (NMA)

All scenarios are emulated using Mininet, a network emulator with limitations in accurately representing complex tactical network scenarios. To address this, we introduced a third auxiliary agent tasked with dynamically manipulating the network links. This approach provides greater flexibility, allowing the simulation of mobility within a TN. In addition, due to link quality changing over time in a TN, we dynamically adjust the bandwidth and delay of each link. This is achieved by leveraging Mininet's built-in functions for editing link parameters. This method enhances the fidelity of the emulation, allowing for more realistic representations of time-varying link quality and network dynamics of heterogeneous tactical networks.

The NMA is implemented using Markov models to simulate different states and transitions of the network. The Markov model consists of five states {0,1,2,3,4} mapped to different values of bandwidth and latency as shown in Fig. 3. The transition between states can be split into the following categories: Remain, improve, and drop. The Markov model is initialized at the beginning of an experiment and stays consistent for the run, introducing broad variation over multiple experiments. This variation yields data that mimics the real world.

TABLE I: Network Topology and Experiment Configuration.

Characteristics	Values	Details
Switches (Vehicles)	10	Each switch forms part of the selected topology
Hosts (Dismounted Units)	20	Distributed equally across switches.
Hosts per Switch	2	Uniform distribution
Attackers	1	Single attacker among the hosts
Controllers	2	Two Ryu SDN controllers one active and one backup.
Topology Types	3	Star, Tree, Linear
Experiments per Topology	10	Each topology undergoes 10 test iterations
Link Quality	Bandwidth: <0.2 Mbit, Latency: 500-2000 ms	Simulates TNs low bandwidth and high latency conditions
Hardware Resources	Linux VM 64-bit, 10 GB RAM, 2 CPUs	Virtualized setup

The initialization of the Markov model follows the following rules: i) Probabilities of remaining in the same state should be consistent and randomized between 0.1 and 0.2; ii) Transitions to neighboring states are more likely; iii) Larger transitions are slightly less likely, following a linear decay function; iv) No transition probability exceeds 0.35; v) The sum of probabilities in each row must equal 1. The state of no connection -an absolute zero- is absent from the defined states because the Mininet link configuration function relies on the Linux `tc` command, which can only set the bandwidth to a minimal value but cannot completely sever the link. The corresponding state transition diagram following all rules is depicted in Fig. 3.

1) Use cases: tactical formations: Switches represent the vehicles of the examined topologies arranged in a star, tree, and linear formation. All other topologies can combine these, except that the Ryu SDN-controller cannot compute cycles in the network. Each previously mentioned topology can be mapped to real-world movement formation as defined in the US field manual 3-90 [18]. A movement formation is an ordered arrangement of forces for a specific purpose. It describes the general configuration of a unit on the ground, allowing a unit to move on the battlefield. There are seven different formations, of which the following six are considered: column, line, wedge, echelon, vee, and diamond. The box formation is omitted, as it maps to a cycle and can, therefore, not be computed. Using standard formations eases a transformation between them, allowing additional flexibility at the Tactical Edge (TE).

IV. Evaluation

A. General settings of the experiment

The experiment is executed in a Mininet environment, running in a virtual machine (VM) on a 64-bit Linux system with two processors and 10 GB of RAM. The controller is deployed using the RYU REST API. Table I lists the complete experiment setup. The topologies are all built with Open vSwitches (vehicles), connected via star, tree, and linear

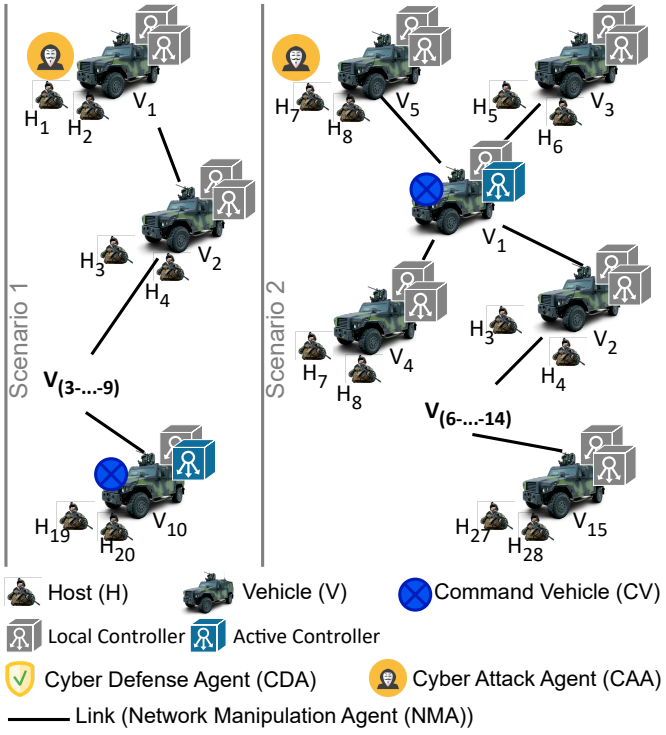


Fig. 4: Scenario (1): Linear topology with highlighted attacker node and (2): Star linear with the attacker node.

topologies. Each represents a different movement formation and is taken from the US field manual 3-90 [18] as mentioned before. Moreover, the NMA creates link variation in the interconnected vehicles through a Markov probability matrix shown in Fig. 3.

In this work, we consider two different scenarios as shown in Fig. 4: (1) a linear topology and (2) a linear star topology (platoons in a convoy). The first topology serves as a base scenario of a single topology, in which the CAA has a local view of the scene and identifies nearby units. This can be mapped to an attacker hiding in a platoon. As these single topologies can be directly constructed from publicly accessible material, such as the US field manual, an attacker can easily train models beforehand. However, these single topologies are only useful in isolated platoons as larger networks consist of multiple single topologies.

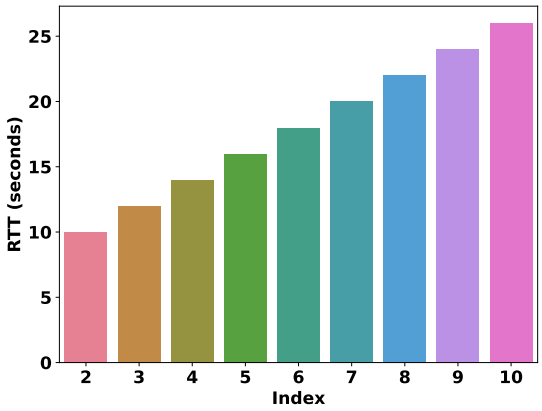
A platoon in a wedge formation connected to a platoon in a Vee formation (linear star topology) is presented in scenario 2, Fig. 4. For these mixed topologies, it is harder to train the model beforehand. Thus, the training on the single topology needs to be as good as possible to enable the model to deal with mixed topologies. Other possibilities of mixed network topologies include platoons connected to a base or multiple platoons connected in various formations. Looking at one possible combination is sufficient, as all other combinations follow the same idea and pattern.

B. Cyber Attack Agent

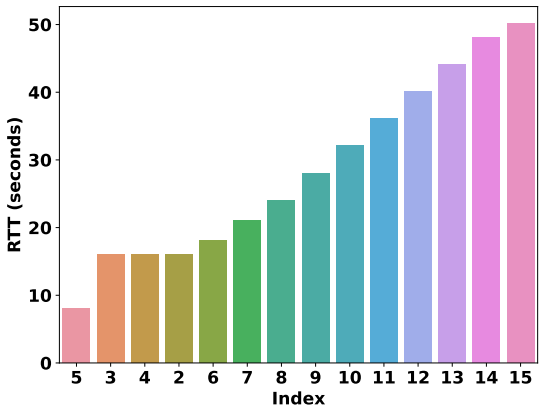
The CAA first goal is to apply network reconnaissance, detecting connected hosts and mapping the information to a network topology using the calculated RTT values. The second goal is to perform a flow table flooding or a DDOS attack.

1) Network Reconnaissance: The evaluation of the network reconnaissance focuses on two aspects. First, the CAA identifies hosts within the network and the protocol used. Second, the effectiveness of classifying the network topology using RTT. Using Nmap allows the CAA to identify all hosts connected to the network. The different protocols tested included ARP, Internet Control Message Protocol (ICMP), Transmission Control Protocol (TCP) SYN, and ping. Due to the restricted links, the important parameters include a minimum RTT to ensure proper execution of Nmap. The CAA only scans the top ten ports to conclude the reconnaissance within a reasonable time. For ICMP, TCP SYN, and ping, we disabled ARP to ensure a different behavior from ARP. All protocols yield the same results. Overusing a certain protocol can alert the CDA. Therefore, the CAA is required to be stealthy. Example RTT values for a linear topology are shown in Fig. 5a and for a mixed topology consisting of linear and star are shown in Fig. 5b.

To evaluate the agent's effectiveness, the CAA applied different models to classify the topologies. To this end, we use the Python library PyCaret, an open-source, low-cost machine-



(a) Linear topology RTT.



(b) Star linear topology RTT.

Fig. 5: RTTs for star linear and linear topology.

TABLE II: Training and Testing Data.

Dataset	Size	Details
Single Topology Set (Training & Testing)	46,094	Linear: 17,442, Tree: 14,440, Star: 14,212
Mixed Topology Set (Evaluation)	9,486	Star linear and Tree linear

TABLE III: Performance on Single and Mixed Topologies.

Model	Accuracy	Recall	Prec.	F1	TT (Sec)
Single Topologies					
ETC	0.7998	0.7998	0.8288	0.8007	0.1710
LGBM	0.7704	0.7704	0.7864	0.7912	0.4110
Mixed Topologies					
ETC	0.5425	0.5425	0.7220	0.5722	-
LGBM	0.5425	0.5425	0.6625	0.5527	-

learning library that automates machine-learning workflows. The training set, see Table II, is the normalized RTT labeled with its corresponding topology (linear, tree, star). The best-performing models are the Extra Tree Classifier (ETC) and Light Gradient Boosting Machine (LGBM), all yielding similar accuracy of ≈ 0.8, recall of ≈ 0.8, precision of ≈ 0.78, and F1-Score of ≈ 0.77. This results in high accuracy in detecting the single type of topology, as shown in Table III.

To challenge the models, we tested them using a testing set with mixed topologies, listed in Table II, that includes a star linear and a tree linear topology. In this case, the model's performance decreases in all metrics: the accuracy dropped to ≈ 0.54, the recall to ≈ 0.54, the precision to ≈ 0.72, and the F1-Score to ≈ 0.57, as shown in Table III. The decrease in performance can be explained by the similarities of the RTT between the different topologies. A major reason is the misclassification of the star topology as linear. If the attacker is positioned in the star topology part, all hosts in this part have the same RTT; however, for every host in the linear part, the RTT increases linearly. This leads the model to group the hosts of the star part to the linear part, as Fig. 5b shows a linear increase.

2) Attack and Evaluation: For a flow table flooding attack, the CAA tries to establish a connection to the victim and measure the time it takes, allowing us to evaluate the stress on the specific switch. Flow Table Flooding attack was configured to send 2,500 packets 60 packets/sec. For the DDOS attack, the CAA tries to establish any new connection within the network, sending 90 packets/sec and measuring the required time, as the controller is the target of the attack. From the CAA point of view, as a result, we noticed a clear increase in the RTT by 62.5%. It is important to note that the strength of the flow table flooding attack has been tuned down to allow the installation of a flow table entry.

C. Cyber Defense Agent

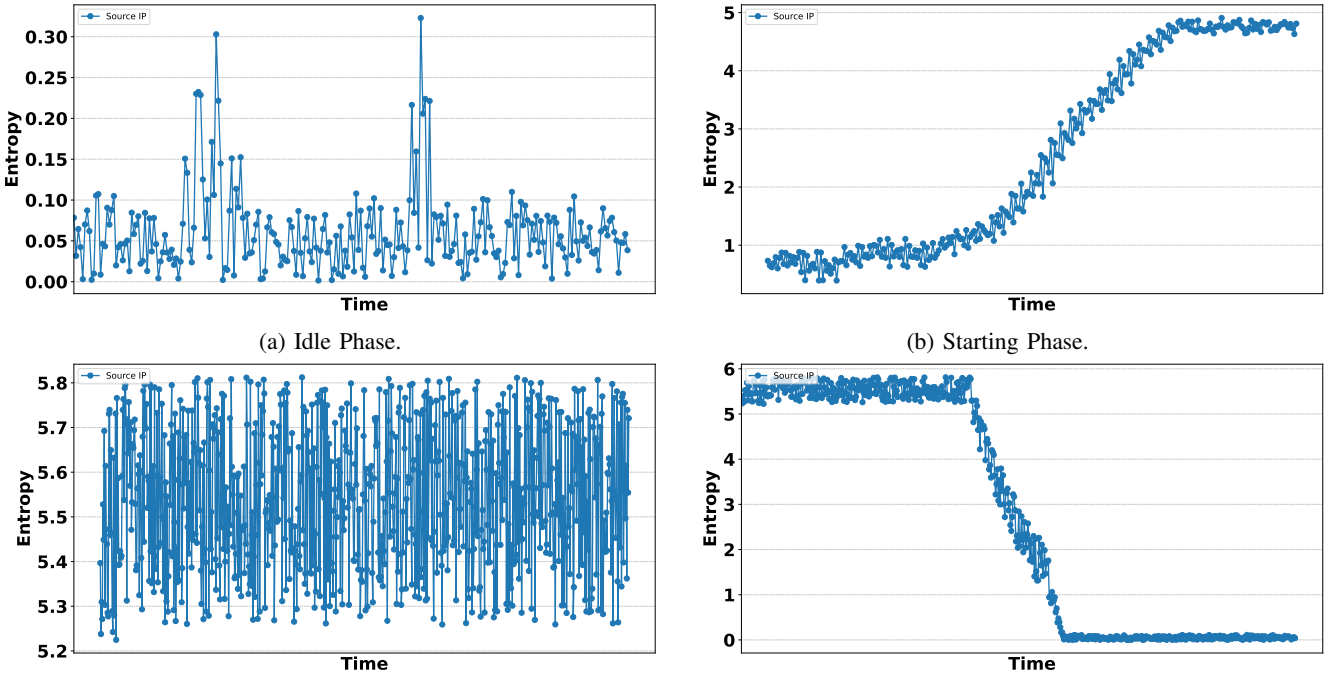
The anomaly detection leverages a learned-based mechanism, extending the threshold-based approach by eliminating thresholds and introducing adaptability. For anomaly detection, the CDA employs two distinct LSTM models, each performing binary classification (anomaly/normal) for a specific attack type: flow table flooding and DDOS attacks based on four attack phases as illustrated in Fig. 6. Using separate models improves the performance metrics by allowing each LSTM to specialize in one task, enhancing accuracy and reducing result ambiguity while maintaining simplicity and robustness.

The key benefits of this split include reduced computational and time costs compared to running a single model with all features. Additionally, the models operate independently, which is crucial for the flow table flooding LSTM, as it must detect attacks quickly due to the potential for rapid damage. The DDOS LSTM, on the other hand, can adapt its execution time windows to optimize computational efficiency and handle unstable network conditions that may cause traffic variations. To identify the optimal model configurations, we use the Python library KerasTuner for scalable and efficient hyperparameter optimization. Both LSTM models share the same architecture, achieving strong results while maintaining implementation simplicity.

The model comprises six layers: input, output, three main, and a dense layer. The model is compiled with the adaptive moment estimation (Adam) optimizer for efficient parameter handling and a tunable learning rate, using binary cross-entropy as the loss function. The time window is set to 1 and defines the observations per step, encompassing all features calculated during the last interval. As feature calculation intervals may vary, no direct mapping exists between the time window and elapsed time. For hyperparameter tuning, random search explores 10 combinations, each tried once, balancing efficiency and search space. The number of epochs is set to 25, with early stopping triggered after five consecutive epochs without validation loss improvement, reverting to the best weights. This approach ensures efficient training and generalization by optimizing validation accuracy.

1) DDOS LSTM: To train the model for detecting DDOS attacks, we define four attack phases (see in Fig. 6) based on changes in IP entropy: 1) *Idle*: Represents normal traffic with low and stable feature values. Spikes occur occasionally with new connections. This phase is critical for defining normal traffic to improve the model accuracy. 2) *Starting:* Features increase continuously, varying from exponential to linear, depending on attack speed and network conditions. Variations can be discrete in unstable networks. 3) *Ongoing:* Feature values stabilize at high levels, with minor variations based on network conditions and attack speed. These values remain distinct from the idle phase. 4) *Ending:* Feature values decrease continuously, mirroring the starting phase. The decrease varies by conditions and ends with idle-phase values.

Unstable network conditions significantly affect entropy changes. Slow traffic throughput can delay entropy increases during attacks, while normal traffic may exhibit large entropy variations in stable networks. The four phases ensure comprehensive training data, enabling robust classification of normal and anomaly states. The data set consists of 94.500 normal



(a) Idle Phase.

(b) Starting Phase.

(c) Ongoing Phase.

(d) Ending Phase.

Fig. 6: Overview of DDOS Attack Phases.

TABLE IV: Overall Metrics and Class Distribution.

Metrics	Value	Class	Size
Accuracy	0.9933	Idle/Normal	94,500
Precision	1	Anomaly (Total)	94,500
Recall	0.9866	End / Start / Ongoing	31,500
F1-Score	0.9932		

samples and 94.500 anomaly samples, with the anomaly samples evenly split into three phases: start, end, and ongoing, with an 80/20 for training and validation. The best model is evaluated on a separate test dataset, representing different scenarios with varying conditions such as bandwidth and delay (see the link quality in Table I), normal and attack traffic speeds, and feature computation intervals. These variations, produced by the NMA, reduce false positives and negatives, enabling the design of a robust model.

Performance is assessed on validation, testing sets, and increasing time windows. Robust models perform well with larger windows, accommodating systems with limited computational capabilities. The model is trained, validated, and tested with a window size of 1 for universal applicability. The DDOS LSTM achieves good validation performance among all metrics, as shown in Table IV. The testing performance is similarly high, demonstrating robustness across varying conditions. These results confirm that the phased approach and feature selection prepare the model effectively for diverse scenarios. Testing with time windows up to 60 seconds reveals a steady decline in performance metrics as the window size increases (see Fig. 7). Accuracy, recall, and F1-score decrease progressively, with recall dropping the most to 0.90, while precision remains stable at 1. The drop in recall indicates misclassification in mixed time windows. In addition, occasional metric anomalies are observed for time windows below 13 seconds, likely caused by the simulation.

2) Flow Table Flooding LSTM: The second LSTM model detects flow table flooding attacks by classifying changes in the number and total entries in the switch. The training data includes 8,000 samples, evenly split between normal and attack scenarios, covering variations in the number of switches flooded and attack intensity. The data is divided by 80/20 for training and validation. These datasets encompass all possible scenarios as unstable network conditions affect only the speed of the attack, and bursts of entries due to instability can be classified similarly to faster flooding attacks. The LSTM model demonstrates strong generalization and balanced performance for both classes as shown in Table IV.

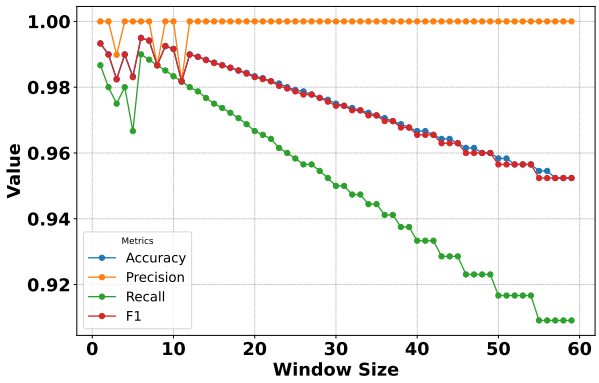


Fig. 7: Metrics over time windows.

3) Threshold-based vs Learned-based Mechanisms: A literature comparison was also made in order to assess the LSTM-based and threshold-based detection systems offer unique advantages for detecting flow table flooding attacks and network reconnaissance. The threshold-based approach presented in [8] is simple, cost-effective, and efficient, requiring only eight thresholds and exhibiting low computational complexity (O(u) and O(u+v)). Where (u) is the number of features (u) and (v) is the number of switches. However, it lacks flexibility, requires detailed environment knowledge for optimal threshold settings, and struggles with scalability and adaptability to changing network conditions. In contrast, the LSTM-based system is adaptable, scalable, and resilient to noise, effectively learning patterns with minimal supervision (human in the loop), provided it has access to high-quality training data. Its drawbacks include high computational costs, dependency on prior training, and the need for a diverse dataset and sufficient controller resources.

V. Conclusion

In this study, we proposed a multi-agent approach to train and test a cyber defense system. The CDA employs a machine-learning-based model for anomaly detection, replacing naive approaches like threshold-based systems. The machine-learning approach uses two LSTM models: one for DDOS attack detection and another for flow table flooding attacks. This separation enhances adaptability, allowing features to be processed at different rates.

The flow table flooding LSTM utilizes changes in switch entries for classification and achieves high accuracy, effectively identifying attacks. DDOS-LSTM is trained on four defined phases, providing the basis for training and validation. Testing on different network scenarios demonstrated high performance, validating its robustness for deployment. Even with varying time windows, the LSTM maintained strong performance, with recall dropping to 0.90 and accuracy and F1-score to 0.95 for a 60-second window, while precision remained stable. Small windows capture rapid changes efficiently but are sensitive to noise, while larger windows identify long-term trends but incur higher computational costs and slower response times.

Future work for the CDA includes enhancing the network topology mapping. Currently, classification supports three categories (tree, star, linear), but performance drops 0.3 across metrics with mixed topologies or additional labels. Incorporating characteristics such as the average and standard deviation of RTT could improve accuracy. Similarly, the flow table flooding LSTM should expand beyond its two-feature reliance to reduce the risk of misclassification.

Acknowledgement

The authors would like to thank The National Council for Scientific and Technological Development – CNPq and Bundeswehr (BAAINBw and WTD 81), and PROPCI-PROPG/UFBA 007/2022 - JOVEMPESQ for supporting the development of this investigation.

References

[1] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27–51, 2014.

[2] P. H. L. Rettore, P. Zissner, M. Alkhowaiter, C. Zou, and P. Sevenich, "Military data space: Challenges, opportunities, and use cases," *IEEE Communications Magazine*, pp. 1–7, 2023.

[3] Rettore, Paulo H. L., J. Mast, T. Aurisch, A. C. Viana, P. Sevenich, and B. P. Santos, "Military iot from management to perception: Challenges and opportunities across layers," *IEEE Internet of Things Magazine*, vol. 8, no. 2, pp. 25–31, 2025.

[4] M. von Rechenberg, P. H. L. Rettore, R. R. F. Lopes, and P. Sevenich, "Software-Defined Networking Applied in Tactical Networks: Problems, Solutions and Open Issues," in *2021 International Conference on Military Communication and Information Systems (ICMCIS)*, 2021.

[5] P. H. Balaraju, P. H. L. Rettore, R. R. F. Lopes, S. M. Eswarappa, and J. Loevenich, "Dynamic adaptation of the user data flow to the changing data rates in vhf networks: An exploratory study," in *2020 11th International Conference on Network of the Future (NoF)*, 2020, pp. 64–72.

[6] P. H. L. Rettore, M. Djurica, R. R. F. Lopes, V. F. S. Mota, E. Cramer, F. Drijver, and J. F. Loevenich, "Towards Software-Defined Tactical Networks: Experiments and Challenges for Control Overhead," in *MILCOM 2022 - IEEE Military Communications Conference (MILCOM)*, 2022.

[7] J. F. Loevenich, P. Zißner, P. H. L. Rettore, J. Bode, T. Hürten, T. Lampe, and R. R. F. Lopes, "Cooperative agent system for quantifying link robustness in tactical networks," in *MILCOM 2023 - 2023 IEEE Military Communications Conference (MILCOM)*, 2023, pp. 411–417.

[8] S. Kloth, P. H. Rettore, P. Zißner, B. P. Santos, and P. Sevenich, "Towards a cyber defense system in software-defined tactical networks," in *2024 International Conference on Military Communication and Information Systems (ICMCIS)*, 2024, pp. 1–8.

[9] P. H. L. Rettore, P. Zißner, M. Lawisch, S. Kloth, E. P. de Freitas, and P. Sevenich, "Towards a resilient multi-agent controller: Securing and mitigating overhead in tactical sdn," in *2024 11th International Conference on Software Defined Systems (SDS)*, 2024, pp. 130–136.

[10] J. A. Perez-Diaz, I. A. Valdovinos, K.-K. R. Choo, and D. Zhu, "A flexible sdn-based architecture for identifying and mitigating low-rate ddos attacks using machine learning," *IEEE Access*, vol. 8, pp. 155 859–155 872, 2020.

[11] M. Li, B. Zhang, G. Wang, B. ZhuGe, X. Jiang, and L. Dong, "A ddos attack detection method based on deep learning two-level model cnn-lstm in sdn network," in *2022 International Conference on Cloud Computing, Big Data Applications and Software Engineering (CBASE)*. IEEE, 2022, pp. 282–287.

[12] D. Tang, Z. Zheng, C. Yin, B. Xiong, Z. Qin, and Q. Yang, "Ftodefender: An efficient flow table overflow attacks defending system in sdn," *Expert Systems with Applications*, vol. 237, p. 121460, 2024.

[13] A. F. Abdullah, F. M. Salem, A. Tammam, and M. H. A. Azeem, "Performance analysis and evaluation of software defined networking controllers against denial of service attacks," in *Journal of Physics: Conference Series*, vol. 1447, no. 1. IOP Publishing, 2020, p. 012007.

[14] K. S. Sahoo, A. Iqbal, P. Maiti, and B. Sahoo, "A machine learning approach for predicting ddos traffic in software defined networks," in *2018 International Conference on Information Technology (ICIT)*. IEEE, 2018, pp. 199–203.

[15] M. Yue, H. Wang, L. Liu, and Z. Wu, "Detecting dos attacks based on multi-features in sdn," *IEEE Access*, vol. 8, pp. 104 688–104 700, 2020.

[16] R. R. F. Lopes, P. H. Balaraju, P. H. L. Rettore, and P. Sevenich, "Queuing over ever-changing communication scenarios in tactical networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 291–305, 2022.

[17] P. H. L. Rettore, J. Loevenich, and R. R. F. Lopes, "TNT: A Tactical Network Test platform to evaluate military systems over ever-changing scenarios," *IEEE Access*, vol. 10, pp. 100 939–100 954, 2022.

[18] H. D. of the Army, "Fm 3-90 tactics," https://irp.fas.org/doddir/army/fm3-90.pdf, accessed: 2024-09-24.