Joint Controller Placement and Intrusion Detection System Enablement in Software Defined Mobile Ad Hoc Networks

Dimitrios Kafetzis and Iordanis Koutsopoulos
Department of Informatics, *Athens University of Economics and Business*, Athens, Greece.

***Abstract*—This work contributes towards improving the security infrastructure in Software-Defined Networking (SDN) enabled Mobile Ad Hoc Networks (MANETs), emphasizing efficient deployment of a reactive firewall. This is a defense system that comprises an SDN controller and several Intrusion Detection Systems (IDSs), and it operates in real-time to detect potential threats and respond to them. IDSs detect network cyber attacks and policy violations, and inform the controller about detected attacks in real-time. We introduce a model and an algorithmic framework for deciding where to place the SDN controller and IDS modules in a network graph. The constraints include a maximum number of IDS modules to place, and a minimum required attack traffic path coverage by IDSs, where a path is said to be covered if there exists at least an IDS placed on a node across the path.**

We propose two algorithms: a Greedy Heuristic and a Simulated Annealing one. Our approaches are evaluated in terms of attack mitigation delay, and they provide IDS deployments with high network path coverage. Validation results using a custom Python simulator and real-world scenarios demonstrate the efficiency of our solutions in comparison to a solution where all nodes are IDS-enabled, and to another one with random IDS modules placement. Our algorithms significantly outperform these baselines in terms of delay and achieve attack path coverage of more than 90%. Additionally, the Simulated Annealing algorithm shows superiority in terms of run-time, and thus it is an attractive candidate for real-world implementations.

***Index Terms*—IDS, Controller Placement, Software Defined Networking, MANETs, Cybersecurity, Networking Optimization.**

I. Introduction

In recent years, the convergence of Software-Defined Networking (SDN) and Mobile Ad Hoc Networks (MANETs) has come to the forefront of the networking industry. Net-centric warfare, a military doctrine where information technology and networked systems are used to gain strategic and tactical advantages, heavily relies on the fusion of SDN and MANETs, exploiting SDN's centralized control and MANETs' inherent adaptability for tactical environments. Yet, this interdependence can also amplify cyber-attack risks. Therefore, efficient countermeasures such as a reactive firewall [1], implemented through an SDN controller and Intrusion Detection Systems (IDSs) like the Snort IDS module [8], are critical. However, the placement of these components in dynamic and resource-limited SDN MANETs presents a significant challenge. Indiscriminate or random placement of IDSs is clearly suboptimal, which the naive solution of placing IDSs on all nodes may turn out to be impractical, given the considerable processing and communication burden they impose. This excess burden can strain the limited capacities of nodes in terms of processing power, memory availability and energy consumption, thus adversely affecting network performance and making this horizontal approach unsuitable for real-time, operational tactical environments.

Therefore, it becomes essential to define a budget, namely a maximum number of nodes on which we will activate IDSs, which refers to a subset of all the MANET nodes equipped to support IDS operation in order to preserve compute and energy resources. This budget consideration is born out of the fact that IDS operation is resource-intensive. Implementing IDS on all nodes can result in substantial energy consumption, which is a critical concern in MANETs given their dependency usually on battery-operated devices.

Determining the optimal subset of nodes on which we will activate IDS instances is a difficult problem because: *i*) it is not known when the attack will be launched, *ii*) the path of the attack traffic is unknown, and *iii*) the extent to which the attack will be detected depends on where we will place the IDSs. Moreover, when the network employs an SDN architecture, the placement of IDSs is interdependent with the placement of the SDN controller because of their cooperation for the reactive firewall implementation. Specifically, the communication delay between the IDSs and the SDN controller, a crucial component of the reactive firewall's response time, highly depends on their relative locations within the network. Thus, optimal IDS and controller placement are critical in order to minimize this delay for a faster and more effective response to potential threats.

To the best of our knowledge, this is the first work that explores jointly IDSs enablement and controller placement within SDN MANETs.

A. *Our contribution*

This paper makes the following contributions:

This work has been carried out in the context of the project entitled, Software defined MObile Tactical Ad hoc NETwork (SMOTANET), which has received funding from the European Defence Industrial Development Programme (EDIDP) under grant agreement No EDIDP-CSAMN-SDN-2019-038-SMOTANET. This paper reflects only the authors' views and the European Commission is not responsible for any use that may be made of the information contained herein.

- We define an optimization problem, the Joint Controller Placement and IDS Enablement (JCPIE) one, applied in an SDN MANET, to minimize the firewall's attack mitigation delay. Key constraints of this problem include:
    i. A budget of a maximum number of concurrently enabled IDS instances in the network.
    ii. A requirement for a minimum attack path coverage by the deployed IDS modules in the network, which ensures that a certain proportion of the possible network communication paths are monitored by IDSs. This is critical because without sufficient coverage, malicious activities will not be effectively detected.
- We propose two algorithms to tackle this class of problem, which is recognized as NP-hard optimization problem. Firstly, we develop a Weighted Greedy Heuristic algorithm that iteratively selects appropriate nodes for IDS placement based on a set of criteria such as node importance and a predefined budget of nodes that can support the IDS operation. The second approach involves a Simulated Annealing algorithm with the same criteria. In our experiments, this latter approach provides faster run-times than the greedy heuristic, while having a similar solution quality, i.e. better mitigation delay.
- We evaluate our proposed algorithmic framework with a methodology founded on a custom-built Python simulator that we develop from scratch, and with artificial and real-world mobility scenarios, which we make publicly available on Github [15]. The proposed algorithms are assessed against a static solution, where all nodes are IDS-enabled, and another solution approach where some randomly selected nodes are IDS-enabled.

The rest of the paper is organized as follows. Section II reviews previous research. In section III, we present the considered model. Section IV presents the problem statement, and section V proposes a centralized solution approach. The evaluation approach, evaluation setup, and results are discussed in section VI. Section VII concludes the paper and gives future work directions.

II. Related Work

Service and IDS placement. Prior research on service placement in MANETs has been focused on either middleware-based or facility location theory-based approaches. The facility location problem in MANETs involves determining the optimal location to place a specific service such as an IDS. Middleware-related approaches employ heuristics based on information gathered from nodes in the neighborhood of the node that hosts the service. In contrast, as it is stated by Wittenberg et al. in their survey [11], facility location theory-based approaches solve the Uncapacitated Facility Location (UFL) problem either with centralized solutions after collecting necessary information from the network or by using distributed iterative approximations. Several approaches have been proposed for IDS sensor placement in static networks. Bhale et al. [10] introduced OPTIMIST, a solution for optimal distributed IDS placement in terms of energy consumption efficiency and capable of mitigating both high and low-rate DDoS attacks in IoT networks. While this work focuses on optimizing IDS placement in IoT networks, our research extends beyond that, by optimizing both the IDS and SDN controller placement in SDN MANETs, thus adding a critical cybersecurity dimension to the controller placement problem. Noel and Jajodia [9] leveraged attack graph analysis for enumerating all potential paths leading to the asset, and subsequently finding the optimal IDS sensor placement to monitor all paths with the fewest sensors possible. Yet, this method neglects crucial factors like traffic load and does not fully encapsulate the intricacies of real-world network environments. In contrast, our research integrates a Simulated Annealing algorithm and a weighted greedy heuristic algorithm, considering the firewall's attack mitigation delay.

Controller placement in SDNs. The objectives of the controller placement problem (CPP) in SDN range from minimizing control latency, improving network resiliency, reducing control overhead to wards energy efficiency. Various optimization models and heuristics have been proposed. For instance, Heller et al. [3] were the first to introduce and formulate the CPP in SDN and proposed three metrics for CPP optimization in its classical basic form, where the objective is to minimize the maximum switch-controller latency. [2] The authors restricted their analysis to Wide-Area Networks (WAN), and evaluated the trade-off between the number of controllers and switch-controller propagation latency. In other works in this direction, Wang et al. [4] proposed a k-means-based network partitioning algorithm to minimize switch-controller latency, while Zeng et al. [5] focused on guaranteeing the flow setup time. Looking to other objectives rather than those addressed by the classical CPP, in order to reduce energy consumption, Hu et al. [6] proposed a binary integer program and a genetic algorithm-based heuristic to minimize the number of links on the switch-controller paths. In terms of the controller's response time, Cheng et al. [7] proposed three heuristic approaches to find the number, location, and switch assignments of controllers, with the objective of meeting an upper bound on the response time of a controller to a switch's flow setup request. These works address the CPP with objectives such as minimizing control latency, they improve network resilience, they reduce control overhead, and they enhance energy efficiency. However, they do not focus attention on cybersecurity aspects, particularly IDS integration and joint consideration with the SDN controller placement problem. In contrast, our work integrates the optimization of IDS and SDN controller placement in the context of SDN MANETs, and thus it adds a crucial cybersecurity dimension to the CPP.

Despite extensive research on the CPP in SDN and generally the facility placement problem regarding facilities like the IDSs, there is no work on the *joint* placement of SDN controller and monitoring services, the latter acting as traffic monitoring sensors and traffic analyzers in a network. In traditional static network architectures, IDSs are deployed in a static manner at strategic points in the network, such as routers or switches, with the aim to monitor traffic for potential security threats. Given the SDN controller's central role in executing our proposed algorithms to solve the JCPIE problem, this approach leverages the unique benefits of SDN's centralized control, thereby paving the way for more efficient solutions of reactive firewall implementations in MANETs.

III. Model

We assume that the network operates in an adversarial environment, where one or more network nodes may act as malicious sources of cyber-threats. In this work, we will focus on DoS attacks.

A. *Cyber-threat assumptions*

We consider that one or more network nodes may execute a malicious service targeting as victim one or more legitimate nodes of the network. The attack can be a DoS flood attack (e.g., a TCP SYN flood attack). As depicted in Fig. 1, the malicious nodes generate traffic directed at a victim node v. It is important to note that the source of flows is not known at the outset, and so it is not known whether a flow is malicious or not. We assume for simplicity that the malicious traffic flows (i.e., flood attack flows) will consistently have much higher data rate than the normal service traffic flows of the network. Any network node can be considered as target of the attack.

B. *Network model*

The network is represented by an undirected graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$ (e.g., the network graph of Fig. 1), where set $\mathcal{N} = \{1, 2, ..., N\}$ denotes all the nodes of the network, and set $\mathcal{E} = \{1, 2, ..., E\}$ denotes the set of the undirected links connecting the N nodes. We assume that one of the nodes is the SDN controller.

Each node $n \in \mathcal{N}$ in the network is associated with a weight w_n, signifying its importance. We assign a high weight to the controller node, considering its crucial role in managing network resources, namely that it maintains a global view of the network, and it makes strategic decisions regarding network configuration and security policies, including the placement of the IDS service. However, this high weight may change with time, contingent on whether the node is selected as the controller or not. As we see later, the proposed algorithms which solve the JCPIE problem (in section V) take these weights as an input to make the selection for IDSs placement. For non-controller nodes, a lower weight is assigned. Hence, while the initial potential weights are inputs to our algorithm, the assigned weights reflect the state of the network and the role of nodes at any given time.

Four main services may be deployed on each node $n \in \mathcal{N}$, (*i*) the SDN switch service that is enabled by default to all nodes, (*ii*) the SDN controller service, (*iii*) the IDS service and (*iv*) a malicious service that launches the attacks. The placement and enablement of the controller and IDS services gives rise to the following binary variables:

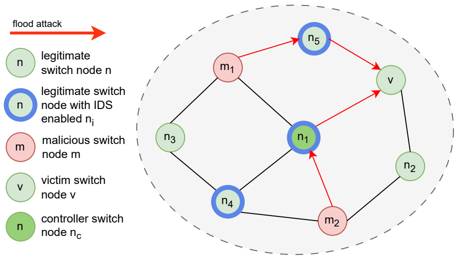


Fig. 1: DoS flood attack in a MANET launched from compromised nodes $m \in \mathcal{N}_m$, targeting as a victim a legitimate node $v \in \mathcal{N}$.

- $c_n \in \{0, 1\}$, where $c_n = 1$ if node n has deployed and enabled the SDN controller service (e.g., Ryu controller), otherwise it is 0.
- $a_n \in \{0, 1\}$, where $a_n = 1$ if node n has deployed and enabled the IDS service (e.g., Snort IDS), otherwise it is 0.

We have the following three subsets of nodes $n \in \mathcal{N}$: $\mathcal{N}_m$, $\mathcal{N}_i$ and $\mathcal{N}_c$ of the set $\mathcal{N}$, where,

- $\mathcal{N}_i \subseteq \mathcal{N}$ is the subset of the SDN switches nodes which have enabled the IDS service;
- $\mathcal{N}_m \subseteq \mathcal{N}$ is the subset of the SDN switches nodes which are compromised and have malicious behaviour;
- $\mathcal{N}_c \subseteq \mathcal{N}$ with $|\mathcal{N}_c| = 1$ is the singleton subset of the SDN switches nodes which has enabled the SDN controller service and has the role of the controller entity in the network.

A switch node n may belong to more than one of the subsets above at the same time, e.g. it can be a controller and a node with deployed IDS.

C. *Intrusion Detection System (IDS) service model*

We take cues from practical IDS solutions such as Snort, operate on a threshold-based system, where traffic that exceeds a preset threshold is flagged as malicious and use this as the rule in our IDS scenario. An IDS monitors the network interfaces of a node, evaluating both inbound and outbound traffic for any malicious patterns through a rule-based recognition system. For instance, a standard Snort rule for TCP SYN flood attacks is designed to closely inspect all TCP traffic, seeking TCP packets that have the SYN flag activated, which indicates the initiation of a TCP connection. A monitoring counter keeps track of each incoming packet's source IP address and increments it with the detection of each TCP SYN packet. If this counter reaches a predefined threshold within a set timeframe, Snort is triggered to send out an alert. Note that, advanced Deep Learning (DL) based techniques may be involved in the traffic's analysis procedure by the IDSs [12], however in this work we focus on simple threshold based rules, in order to showcase the joint problem of the IDS deployment and controller placement problem and the solutions we propose.

Since only a limited number of nodes can support IDS operation, there exists partial coverage of the communication paths of the network by the IDSs, meaning that the activated IDSs monitor only a subset of the set $\mathcal{P}$ of all possible paths. Each IDS can monitor one or more paths and at least one IDS service is required to reside on a node across a path so that the path is considered to be covered by the IDS. Let us introduce a binary parameter p_{in} which equals 1 if path i passes through node n, i.e. the node n monitors the path i, and 0 otherwise. The minimum attack path coverage percentage, P_{min} indicates the minimum proportion of the network's paths that must be under surveillance by the IDSs to ensure an acceptable level of security. A representative acceptable attack path coverage percentage can be around $80\% - 90\%$. In the absence of any prior knowledge on attack deployment, it is impossible to predict the precise route an attack will take, and thus we have to consider all possible paths the traffic could traverse. Therefore, the constraint for a minimum required attack path coverage percentage P_{min} can be expressed as follows:

$$\frac{\sum_{i \in \mathcal{P}} \max_{n \in \mathcal{N}} (a_n \cdot p_{in})}{|\mathcal{P}|} \geq P_{min}, \tag{1}$$

where p_{in} is a binary parameter equal to 1 if path i is monitored by an IDS on node n, and 0 otherwise. Here, $a_n \cdot p_{in}$ will be 1 if a node n is on path i and has an enabled IDS and 0 if no nodes on the path has enabled IDS. Thus, the numerator counts the number of covered paths.

Upon receiving the IDS alerts, the SDN controller establishes countermeasures against compromised network nodes i.e., black-listing of malicious nodes and setup of blocking flow rules at all nodes of the network. It also orchestrates the dynamic adjustment of the IDS state i_n at network nodes.

1) Budget-based IDS deployment in MANET: Resource constraints in MANETs mandate a judicious, budget-based IDS deployment. Besides computaitonal resources, the communication links' overhead—measured by bandwidth used for non-data tasks such as control messages, IDS alerts, and operational data—is a critical factor. Each additional IDS instance exacerbates this overhead with alert messages generated upon threat detection, like flood attacks. These alerts, though small, can have a significant cumulative impact on the constrained bandwidth during sustained attacks, straining both computational resources and network links. For instance, popular IDSs like Snort can generate alerts of the order of hundreds of Kbps under heavy load, thus significantly impacting the limited wireless bandwidth of MANETs. Moreover, as the network expands, so does the number of paths needing surveillance, potentially increasing the need for coverage by IDSs beyond what nodes and links can handle. Thus, a minimum budget-based IDS deployment adjustment becomes crucial.

The budget is defined as the number of nodes B that are needed so as to support the IDS operation in order to ensure a minimum attack path coverage percentage P_{min} of network paths by the IDSs' monitoring. In our scenario, network paths represent the set of elements to be covered. Placing the IDS on a node corresponds to monitoring a *subset* of paths, those that pass through the node that has the IDS. Thus, finding the value of B is reminiscent of a set-covering problem, where the goal is to cover at least the minimum percentage P_{min} of network paths using the minimum number of nodes. Since set-covering problems are known to be NP-hard, in order to find a feasible solution to the set-covering problem, we employ a Greedy Heuristic Algorithm. The steps of the algorithm are as follows:

```
Step 1. Initialize set $\mathcal{B} = \emptyset$, and create a copy of the set of all network paths $\mathcal{P} = \mathcal{P}_{\text{copy}}$.
Step 2. While the coverage of $\mathcal{P}_{\text{copy}}$ is less than $P_{min}$, perform the following steps:
    Step 2.1 For each node $n \in \mathcal{N}$, calculate the size of the set of network paths, $\mathcal{P}_n$, that can be covered by enabling the IDS on node $n$.
    Step 2.2 Select the node $n' = \arg\max_{n \in \mathcal{N}} |\mathcal{P}_n|$ that covers the maximum number of uncovered network paths.
    Step 2.3 Add node $n'$ to $\mathcal{B}$ and remove the paths covered by $n'$ from $\mathcal{P}_{\text{copy}}$: $\mathcal{B} = \mathcal{B} \cup \{n'\}$ and $\mathcal{P}_{\text{copy}} = \mathcal{P}_{\text{copy}} \setminus \mathcal{P}_{n'}$.
Step 3. The output $\mathcal{B}$ is a feasible solution with the set of nodes where the IDS must be enabled to achieve a coverage of at least $P_{min}$.
```

IV. The Joint Controller Placement and IDSs Enablement (JCPIE) Problem

A fundamental objective of a cybersecurity framework is to rapidly react to cyber-attacks. In our case, this is performed with the aid of a reactive firewall implemented by the SDN controller and the set of activated IDSs. The operation of the firewall has 3 phases, as depicted in Fig. 2 (a) : (*i*) the *attack detection* phase, where one or more IDSs detect an attack at the network, (*ii*) the *controller alerting* phase, where the IDSs inform the SDN controller about the attacks' detection through alert messages, and (*iii*) the *black-listing* of the malicious nodes that the controller performs by sending flows' setup messages to the nodes of the network. As depicted in Fig.2 (b), the two main procedures of the firewall's operation, the *Alerting* and the *Black-listing* create significant delays:

- $D_{alerting}$, the alerting delay, is the time delay from the moment when an enabled IDS on a node n (where $a_n = 1$) detects an attack, until the alert message sent by this IDS arrives at the SDN controller, denoted as node c (where $c_n = 1$).
- $D_{black-listing}$, the black-listing delay, is the time delay from the point in time when the controller receives the alert message, until all nodes in the network receive the flow setup message sent by the controller for black-listing the malicious node.

The attack mitigation delay $D_{mitigation}$ is the sum of the alerting and black-listing delays, i.e. $D_{mitigation} = D_{alerting} + D_{black-listing}$. This sum captures the critical interplay between IDSs and controller placement. Specifically, the alerting delay $D_{alerting}$ is impacted by the positioning of IDSs as it is contingent on the propagation time of an alert from an IDS to the SDN controller. Also, the black-listing delay $D_{black-listing}$

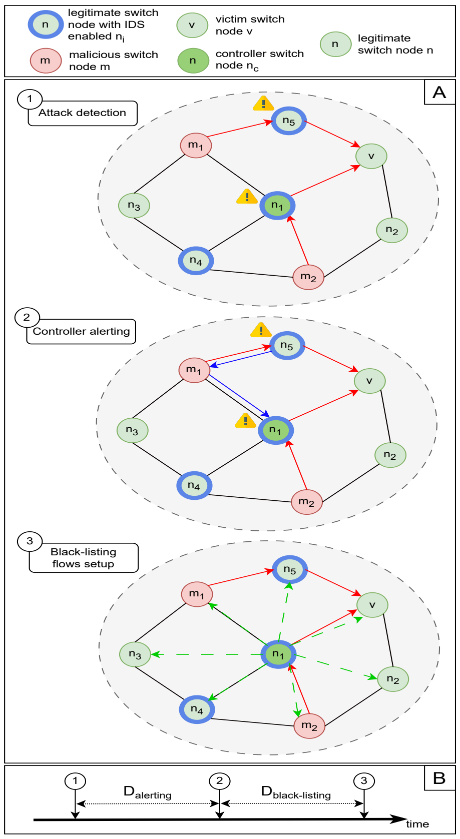


Fig. 2: (a): The three phases of reactive firewall operation of the considered scenario: 1) Attack detection (red arrows: the attack flows pass through the IDSs), 2) Controller alerting (blue arrows: the IDSs send alert messages to the controller) and 3) Black-listing flows setup (green arrows: the controller sends messages to all nodes informing them about which flows are attack flows), (b): The mitigation delay $D_{mitigation}$ components, $D_{alerting}$ and $D_{black-listing}$.

is affected by the placement of the SDN controller. So, in order to minimize mitigation delay, a joint consideration of both IDS and controller placement is needed.

We define an average latency $\bar{\tau}$ per hop in the network, and the distance between any two nodes n and n' in number of hops to be $d(n, n')$. Then we can express $D_{alerting}$ as follows:

$$D_{alerting} = \sum_{n,n' \in \mathcal{N}} a_n \cdot c_{n'} \cdot \bar{\tau} \cdot d(n, n'), \tag{2}$$

where, the delay $D_{alerting}$ is computed for each pair of nodes (n, n') in the network. The term $a_n \cdot c_{n'}$ serves as a condition that the delay is only accounted for when node n has an IDS enabled ($a_n = 1$), and node n' is the controller node ($c_{n'} = 1$). Further, it is

$$D_{black-listing} = \sum_{n,n' \in \mathcal{N}} c_n \cdot (1 - c_{n'}) \cdot \bar{\tau} \cdot d(n, n'), \tag{3}$$

Similar to the alerting delay, this formula sums up the delays between each pair of nodes (n, n') in the network. The term $c_n \cdot (1 - c_{n'})$ is a condition that signifies the delay is only accounted for when node n is the controller ($c_n = 1$) and node n' is any other node except the controller ($c_{n'} = 0$).

We define the Joint Controller Placement and IDS Enablement (JCPIE) problem as the problem of placing attack monitoring modules (e.g. IDSs) and a single SDN controller so as to minimize the attack mitigation delay of the reactive firewall $D_{mitigation}$ over the mitigation delays of all IDS-enabled node-controller pairs subject to a budget B of IDS-enabled nodes. The problem can be formulated as follows:

$$\begin{aligned} \min_{c_n, a_n \forall n} \quad & \frac{1}{\sum_{n \in \mathcal{N}} a_n} D_{mitigation} \\ \text{s.t.} \quad & \sum_{n \in \mathcal{N}} a_n \leq B \\ & \sum_{n \in \mathcal{N}} c_n = 1 \\ & \sum_{n \in \mathcal{N}} w_n \cdot c_n \geq w_{max} \\ & \frac{\sum_{i \in \mathcal{P}} \max_{n \in \mathcal{N}} (a_n \cdot p_{in})}{|\mathcal{P}|} \geq P_{min} \\ & c_n \leq i_n \quad n \in \mathcal{N} \\ & c_n \in \{0, 1\} \quad n \in \mathcal{N} \\ & a_n \in \{0, 1\} \quad n \in \mathcal{N} \end{aligned} \tag{4}$$

The solution to the JCPIE problem as described has two outputs: i) a singleton subset $\mathcal{N}_c$ that identifies the selected SDN controller, and ii) a policy represented as a binary vector $(a_1, ..., a_N)$, where elements that are equal to 1 indicate nodes where IDSs are placed. These decisions adhere to several constraints. Firstly, the total count of IDS-enabled nodes must not exceed the given budget, B. Secondly, there must be a single SDN controller in the network, which must be the node with the highest weight. This is guaranteed by the constraint $\sum_{n \in \mathcal{N}} w_n \cdot c_n \geq w_{max}$, reflecting the controller's crucial role. Importantly, the controller node, due to its high weight, must also be under IDS surveillance, denoted by the constraint $c_n \leq i_n$. Lastly, the percentage of IDSs coverage over the paths of the considered MANET, represented as $\frac{\sum_{i \in \mathcal{P}} \max_{n \in \mathcal{N}} (a_n \cdot p_{in})}{|\mathcal{P}|}$, must meet or exceed the minimum required attack path coverage, P_{min}, which is captured by the last constraint.

V. Centralized Solution Approaches for the JCPIE Problem

In this section, we propose centralized solution approaches to tackle the JCPIE problem. The use of SDN allows for greater flexibility and centralized control, enhancing the ability to place and enable IDS instances across the network. The controller has a global view of the network, and thus it collects necessary information from the network, it decides on the IDS placement, and it returns commands to the appropriate network nodes to activate their IDSs. Two distinct approaches are proposed in this work - the Weighted Greedy Heuristic (WGH) and the Simulated Annealing (SA) algorithms. The SDN controller executes the algorithms periodically, with the period reflecting the network's mobility dynamics.

A. Weighted Greedy Heuristic Algorithm

First, we propose a two-stage WGH algorithm, which is a scalable approach and ideal for dynamic networks like MANETs. This algorithm leverages the inherent simplicity of greedy methods to reduce the attack mitigation delay of the reactive firewall. The algorithm consists of two stages: 1) SDN controller placement and 2) IDS enablement.

Stage 1: SDN Controller Placement

The first stage of the algorithm focuses on finding a good location for the SDN controller, based on minimizing the delay from the SDN controller node to other nodes. It has the following steps:

Step 1. Each node n in the network periodically exchanges messages with all other nodes n' to measure the average delay $d(n, n')$. Once the measurements are collected by all nodes, each node then sends the values of these delays to the current SDN controller.

Step 2. Initialize an empty set $\mathcal{C}$ to hold the candidate nodes for the SDN controller.

Step 3. For each node $n \in \mathcal{N}$, calculate the total sum of average delays to all other nodes n', $\bar{\tau} \cdot d(n, n')$.

Step 4. Determine the node n with the minimum sum of delays, set as $n^* = \arg\min_{n \in \mathcal{N}} \bar{\tau} \cdot \sum_{n'} d(n, n')$. Mark this node as the SDN controller by setting $c_n = 1$ and assign the SDN controller a higher weight than the other nodes. And the remaining nodes are assigned a lower weight, equal for all of them. These weights are assigned before the algorithm runs.

Stage 2: IDS Enablement

The second stage of the algorithm solves the IDS enablement problem that reduces $D_{mitigation}$ while staying within budget B that is the required number of nodes so as to support the IDS operation in order to ensure a minimum attack path coverage percentage P_{min} of network paths by the IDSs' monitoring.

Step 1. Start with an empty set $\mathcal{N}_i$ for nodes with enabled IDS.

Step 2. While the budget B is not exceeded, perform the following:

- Step 2.1 For each node $n \in \mathcal{N} \setminus \mathcal{N}_i$, calculate the potential reduction in mitigation delay $\Delta D_{mitigation}$ if the IDS is enabled on node n.
- Step 2.2 Select the node n' that leads to the highest possible product of node weight and reduction in $D_{mitigation}$.
- Step 2.3 If by enabling the IDS on node n', the budget B is not exceeded, then update $\mathcal{N}_i = \mathcal{N}_i \cup \{n'\}$, and set $a_{n'} = 1$.
- Step 2.4 Compare the current attack traffic path coverage to the minimum requirement P_{min}. If coverage has not reached the P_{min}, continue to the next iteration.
- Step 2.5 Terminate the algorithm if B is exceeded or if the coverage requirement P_{min} is satisfied.

The solution to the WGH algorithm gives the location of the SDN controller and the IDS enablement policy. The algorithm is executed periodically through the SDN controller, which collects network topology data, constraints, and requirements. In MANETs the controller node may change, necessitating a state transfer from the old controller to the new. An alternative, Simulated Annealing algorithm-based approach, which mitigates issues with execution run-time, especially for large number of nodes, is discussed next.

B. Simulated Annealing Algorithm

The SA algorithm provides a more scalable solution to the JCPIE problem compared to the WGH one due to its ability to explore a larger solution space through the iterative refinement of temporary suboptimal solutions, facilitating enhanced optimization especially in larger networks. The SA algorithm [14] is a stochastic optimization technique that draws inspiration from the annealing process in metallurgy, where a material is slowly cooled to minimize its defects, thus achieving an optimal state. The SA algorithm works as follows:

Step 1. Start by assigning initial values to the *temperature* parameter T and the *cooling rate* parameter α. In the SA context, *temperature* T is a tunable parameter that controls the acceptance of solutions that do not minimize immediately the mitigation delay.

Step 2. Start with an initial solution where IDS nodes are randomly placed in the network, not exceeding budget B and ensuring P_{min}, including a randomly selected controller node.

Step 3. Repeat the following process until $T \leq 1$:

- Step 3.1 Generate a new candidate solution by randomly relocating an IDS node, ensuring that it does not violate the budget B and the minimum attack path coverage constraints.
- Step 3.2 Compute mitigation delay $D_{mitigation}$ for the new solution.
- Step 3.3 If the new solution improves the mitigation delay, then accept it. Otherwise, accept it with a probability $P(\text{accept}) = \exp\left(-\frac{\Delta D_{mitigation}}{T}\right)$. This *cooling schedule*, based on the concept of temperature in simulated annealing, allows acceptance of suboptimal solutions to avoid local optima.
- Step 3.4 Update the temperature T using the cooling rate α via $T = T(1 - \alpha)$. The process of gradually reducing T simulates the *cooling* phase in the annealing process, allowing the algorithm to eventually focus on the exploration around a good solution [14].

Step 4. Finally, return the solution that resulted in the lowest $D_{mitigation}$ during the search.

The acceptance probability function at the SA algorithm's step 3.3 is based on the Boltzmann distribution [14], which is fundamental to the simulated annealing process, allowing occasional acceptance of worse solutions to avoid local optima. Also, the stopping criterion of $T \leq 1$, was chosen as it strikes a balance between achieving a sufficient number of algorithm iterations for exploring the solution space and ensuring computational efficiency, as determined through empirical testing that we performed.

The SA algorithm offers several advantages over the WGH algorithm. It is capable of finding good solutions in larger solution spaces, which is essential for larger networks, by avoiding getting trapped in local optima. Also, due to its stochastic nature, the SA algorithm is faster and can be parallelized, leading to further speed up in the optimization process. Nevertheless, the SA algorithm's performance can be optimized by carefully tuning its hyperparameters, particularly the initial temperature T and the cooling rate α, so as to enable a wider solution space exploration and more iterations for convergence, respectively. This is particularly useful in larger, more complex networks.

VI. Evaluation

A. Evaluation setup

To evaluate the solutions to our proposed JCPIE problem, we developed a custom Python-based MANET simulator which we make available on GitHub [15]. The simulator allows user-defined network topology parameters such as the number of nodes and their mobility mode. Also it can execute mobility scenarios using Network Simulator (NS) movement files that contain coordinates representing nodes' movements. We consider three performance metrics:

i. *average attack mitigation delay*,
ii. *average attack path coverage percentage*, and
iii. *average run-time of the algorithms*.

All of these metrics are averaged over 50 mobility frames where the MANET topology changes. Networks ranging from 5 to 18 nodes are evaluated under two mobility scenarios:

i. An *artificial* mobility scenario, where nodes move randomly across a 1000 x 1000 m^2 area, with varying speeds and directions in each frame. This scenario serves to stress-test the algorithms' adaptability to rapid network topology changes within each execution cycle.
ii. A *real* mobility scenario based on the Anglova [13] tactical scenario during engagement phase, where nodes split into small teams moving randomly in the terrain. This scenario provides varying network topologies that mirror real-world conditions, allowing us to evaluate the algorithms' potential performance in actual operational environments.

We evaluated our proposed WGH and SA algorithms, using a 2-minutes execution cycle at the SDN controller node. The execution period is set in the order of minutes, tuned to the medium-speed nodes' movement of the considered scenarios, thus striking a balance between adaptivity to network topology changes and computational efficiency. Performance comparisons were made against two benchmark static solutions:

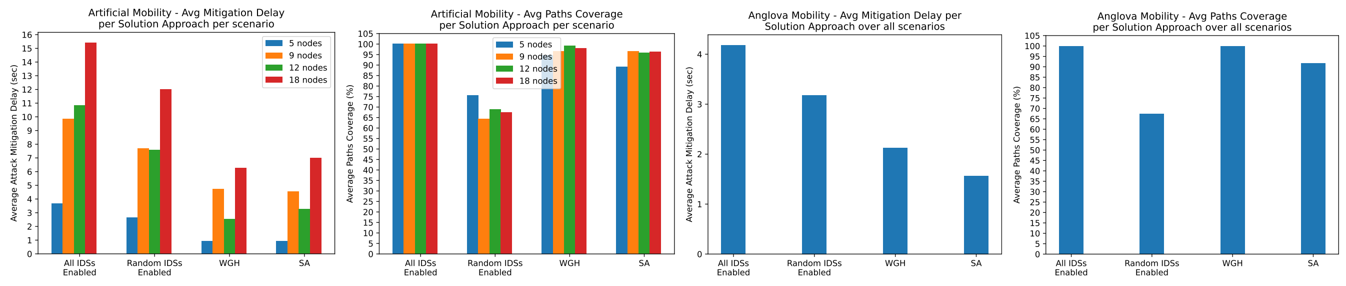
i. *All IDSs Enabled*, i.e. all nodes operating an IDS service.
ii. *Random IDSs Enabled*, i.e. we arbitrarily enable IDSs on 80% of nodes, chosen uniformly at random, with each node having an equal chance of selection.

B. Performance Evaluation Results

Our evaluation results illustrate the average attack mitigation delay of the firewall (Fig.3 (a).1, (b).1), the average attack path coverage percentage by the IDSs' surveillance (Fig. 3 (a).2, (b).2), and the proposed algorithms' execution run-time (Fig. 4) for each scenario. In all scenarios, the proposed WGH and SA algorithms consistently outperform the solution where a random subset of 80% of the nodes have IDS enabled. Also, the random IDS activation solution fails to achieve the 80% attack coverage threshold in any of the evaluated scenarios, with its performance ranging between 65% and 71% attack path coverage across all scenarios. Our proposed algorithms consistently maintain a value of attack path coverage by the IDSs which is above the 80% threshold across all paths, surpassing the 95% attack path coverage on average, and performing much better than the *Random IDSs Enabled* approach. Additionally, both the WGH and SA algorithms perform better than the *All IDSs Enabled* static solution approach in terms of average mitigation delay reduce. As expected, the dynamic algorithms excel in attack mitigation delay reduction, unlike the *All IDSs Enabled* solution which focuses solely on achieving full network paths coverage by the IDSs.

1) Lessons learned and key insights:

- Our proposed SA and WGH algorithms outperform both the compared solution approaches (*All IDSs Enabled* and *Random IDSs Enabled*) in terms of both the reduce of attack mitigation delay, while keeping a path coverage higher than the threshold in almost all scenarios. This demonstrates their ability to adapt efficiently to network changes, while improving the reactive firewall's response time.
- The WGH algorithm consistently outperforms all benchmark solution approaches in terms of attack mitigation delay of the reactive firewall in the artificial mobility scenario. However, in the Anglova scenario, the SA algorithm slightly outperforms the WGH one by 250 msec on average in terms of attack mitigation delay across all scenarios. This improvement is likely due to the SA's probabilistic nature, which allows it to explore a wider range of potential solutions and thereby better adapt to network dynamics.
- The WGH algorithm achieves almost 100% coverage over the network paths at all scenarios (Fig. 3 (a).2, (b).2). In contrast, the SA algorithm achieves slightly lower coverage by 7% than the WGH algorithm's achieved one.
- The SA algorithm is faster in terms of execution run-time than WGH, which is a crucial attribute for real implementations of the firewall.



a.1 a.2 b.1 b.2

Fig. 3: **(a) Artificial random mobility**: a.1) Average attack mitigation delay (in sec) for different IDS placement approaches and different network sizes. a.2) Average attack path coverage (%) by IDSs for different IDS placement approaches and different network sizes. **(b) Anglova scenario mobility**: b.1) Average mitigation delay (in sec) and b.2) Average attack path coverage (%) by IDSs, both for different IDS placement approaches and different network sizes.

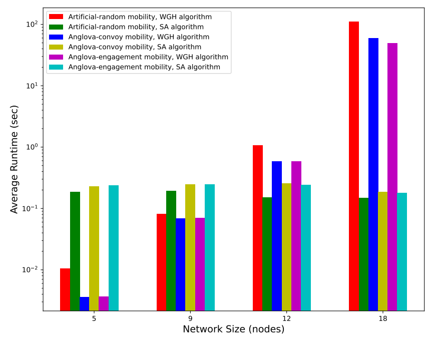


Fig. 4: A comparison between the WGH (red, blue and purple bars) and SA algorithms (green, yellow and cyan bars) algorithms regarding their **average execution run-time** (in sec) per mobility scenario evaluated in 4 network sizes (5, 9, 12 and 18 nodes).

VII. Conclusion and Future Work

In this work, we addressed the joint problem of the placement of IDSs and of an SDN controller in SDN MANETs so as to reduce attack mitigation delay of the deployed reactive firewall (i.e., comprised by IDSs and controller) in dynamic and resource-limited environments, like tactical networks. We proposed two algorithms - a greedy one, and one based on Simulated Annealing. Both algorithms ensure a certain required attack path coverage while outperforming two other standard IDS-enabled solutions in the evaluation metrics. Additionally, the SA approach gives better run-time performance than the greedy one. However, our algorithms currently face scalability issues. As the network size expands, we note a corresponding increase in both attack mitigation delay and execution run-time, a trend confirmed by our evaluation results, highlighting areas for future improvements.

In our pursuit of scalable solutions, we intend to explore the potentials of a decentralized approach in the next phase of our research. This approach stands to significantly diminish computational load and time complexities which currently exhibit linear or superlinear growth with increasing network size. Utilizing decentralized algorithms, we anticipate leveraging localized information, thereby expediting responses and adaptations to dynamic network conditions. This direction aims to enhance the scalability of our security solutions, fostering a robust defense mechanism that maintains its efficiency even as network size expands significantly.

References

[1] S. Zerkane et al., “Software defined networking reactive stateful firewall,” ICT Systems Security And Privacy Protection: 31st IFIP TC 11 Int. Conf., SEC 2016, Ghent, 2016.
[2] T. Das et al., “A Survey on Controller Placement in SDN,” IEEE Communications Surveys & Tutorials, vol. 22, no. 1, pp. 472-503, First quarter 2020.
[3] B. Heller et al., “The controller placement problem,” Proc. ACM 1st Workshop Hot Topics Softw. Defined Netw., 2012.
[4] G. Wang et al., “A K-means-based network partition algorithm for controller placement in software defined network,” Proc. IEEE Int. Conf. Commun. (ICC), 2016.
[5] D. Zeng et al., “Flow setup time aware minimum cost switch-controller association in software-define networks,” Proc. IEEE 11th Int. Conf. Heterogeneous Netw. Qual. Rel. Security Robustness (QSHINE), 2015.
[6] Y. Hu et al., “The energy-aware controller placement problem in software defined networks,” IEEE Commun. Lett., vol. 21, no. 4, pp. 741-744, April, 2017.
[7] T. Y. Cheng et al., “QoS-guaranteed controller placement in SDN,” Proc. IEEE Glob. Commun. Conf. (GLOBECOM), 2015.
[8] M. Roesch, “Snort - lightweight intrusion detection for networks,” Proc. of the 13th USENIX Conf. on System Administration (LISA’99), 1999.
[9] S. Noel and S. Jajodia, “Attack graphs for sensor placement, alert prioritization, and attack response,” Cyberspace Research Workshop, 2007.
[10] P. Bhale et al., ”OPTIMIST: Lightweight and Transparent IDS With Optimum Placement Strategy to Mitigate Mixed-Rate DDoS Attacks in IoT Networks,” in IEEE Internet of Things Journal, vol. 10, no. 10, pp. 8357-8370, May 2023.
[11] G. Wittenburg and J. Schiller, “A Survey of Current Directions in Service Placement in Mobile Ad-hoc Networks,” IEEE Int. Conf. on Perv. Comp. and Commun. (PerCom), 2008.
[12] M. S. ElSayed et al., “A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique,”Journal of Network and Computer Applications, vol 191, 2021.
[13] N. Suri et al., “The angloval tactical military scenario and experimentation environment,” 2018 Int. Conf. on Military Communications and Information Systems (ICMCIS), 2018.
[14] S. Kirkpatrick et al., “Optimization by Simulated Annealing,” Science, vol 220, pp. 671-680, 1983.
[15] The JCPIE simulator, https://github.com/Dimitrios-Kafetzis/SDN-MANET-reactive-firewall-simulators , July 2023.