Cybersecurity Fusion: Leveraging Mafia Game Tactics and Reinforcement Learning for Botnet Detection

Amir Javadpour*§, Forough Ja'fari†, Tarik Taleb*, HamidReza Ahmadi‡, and Chafika Benzaïd*
*Faculty of Information Technology and Electrical Engineering, University of Oulu, Oulu, 90570, Finland
§ICTFICIAL Oy, Espoo, Finland
†Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
‡Faculty of New Sciences and Technologies, University of Tehran, Tehran, Iran

***Abstract*—Mafia, also known as Werewolf, is a game of uncertainty between two teams, which aims to eliminate the other team's players from the game. The similarities between detecting the Mafia members in this game and botnet detection in a computer network motivate us to solve the botnet detection problem using this game's winning strategies. None of the state-of-the-art researches have used the Mafia game strategies to detect the network's malicious nodes. In this paper, we first propose the Mafia detection strategies, which are applied using linear relation and reinforcement learning techniques. We then use the suggested strategies in a network infected by the Mirai botnet, using Mininet, to evaluate the performance of botnet detection. The average results show that the suggested strategies are 11% more accurate than the existing ones for the Mafia game. Additionally, the true positive and true negative detection rates of a network modeled by the proposed Mafia game are 71% and 91%, respectively.**

***Index Terms*—Mafia game, Reinforcement learning, Network security, Botnet detection, Distributed denial of service (DDoS) attacks, and cybersecurity.**

I. Introduction

The diminution in the functionality of a network after a cyber attack, especially the distributed ones, is undeniable. Some adversaries build up an army using the compromised nodes and then launch a distributed attack against the valuable assets in the network. This army is called a botnet, and since its members are distributed, their attack is hard to detect. The botnets infect billions of devices, and mitigating these threats is highly important, especially in large-scale networks such as the Internet of Things (IoT) [1, 2, 3].

The researchers have proposed several botnet detection techniques. The most recent ones in different categories are as follows. Ashraf et al. [4] and Popoola et al. [5] used learning approaches to detect bots based on their behavioral patterns. Ja'fari et al. [6] used deception techniques to lure the bots and detect them. Joshi et al. [7] utilized fuzzy techniques for training a neural network to detect the bots. Abu Khurma et al. [8] proposed a botnet detection technique based on optimization algorithms.

However, none of the researches in this field have focused on the bots' impact on the networks during their lifecycle to detect them. For example, suppose there are hundreds of nodes in a network, one-third of which are compromised as bots. A Distributed Denial of Service (DDoS) attack is launched against one of the critical servers in this network [9]. Among those who have connected with this server, some are legal nodes, and others are bots [10, 11, 12]. The state-of-the-art botnet detection techniques first detect the bots and then block them. However, because the current detection techniques do not have a true positive rate of 100%, all the bots are not detected. The remaining ones can again launch a DDoS attack against another server in the next attack phase. Our goal is to propose a way of detecting the bots based on their impact in the previous attack phases.

Mafia is a role-based party game, invented by Dimitry Davidoff, that includes two teams of players: Mafias and Townies. The goal of each team is to eliminate the players of the other team from the game. The Townies do not know each player's role, so they try to find clues about the Mafias. On the other hand, the Mafias know each other, but they attempt to lure the Townies by pretending to be a Townie. A voting process is performed at the end of each phase of the game. The players vote to choose the one who must exit the game. The player receiving the most votes is removed from the game. Different scenarios are defined for this game. Two extra roles are defined in one of the most simple ones: Detective and Proof. The Detective is a Townie who can inquest one of the players, and realize whether it is in the Mafia team [13]. The Detective's challenge is which player is likely to be a Mafia. The Proof in a Mafia game is one of the Townies, who never leaves the game. The role of the Proof is hidden from the other players, but once it receives the highest number of votes, its role becomes revealed. Finding the Mafia members is based on their votes/impact in the game, which is similar to our goal in detecting the bots.

The current researches on the Mafia game has tried to give a detection strategy. Chang [14] proposed a simulator that assigns a credibility weight to each of the players, by which the powerful Townie players are found. Kondoh et al. [15] used long short-term memory learning to design an efficient agent for the Mafia game. The features used for training the agent are the conversation between the players and their votes for all the other players. Hagiwara et al. [16] also considered the conversations and the votes to train a Mafia agent that utilizes reinforcement learning. Bi and Tanaka [17] suggested the Detective randomly requesting the players. We call this

979-8-3503-1090-0/23/$31.00

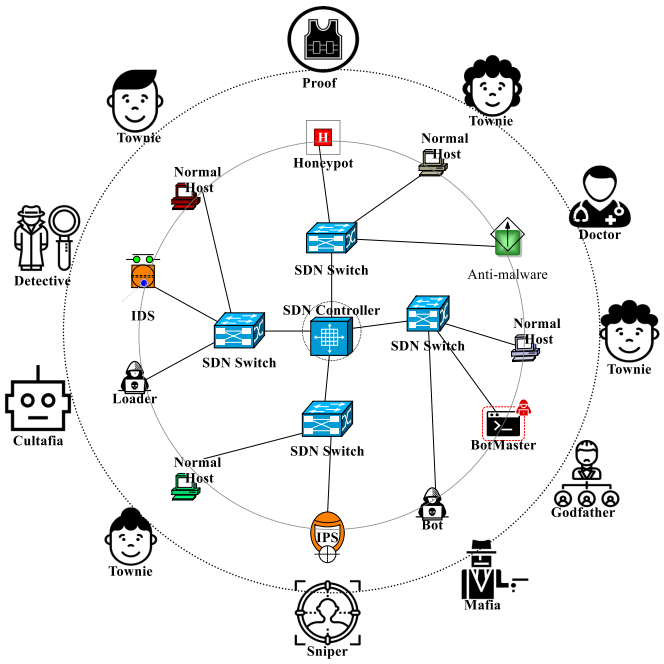


Fig. 1. The mapping between the Mafia roles and the components of a network under a botnet attack.

strategy as *"Previous Strategy 1"*. As another example, Wang and Kaneko [18] proposed a machine learning model that selects the player for being requested by three parameters, including (1) "the number of each player's votes to each of the other players", (2) "the state of being active or inactive for all the players", and (3) "the previous actions of the Detective". We call this strategy as *"Previous Strategy 2"*. However, these strategies are not powerful enough to make essential detection decisions. Moreover, none of them used these strategies to detect malicious nodes in a computer network.

The main goal of this paper is to model the process of botnet detection as detecting the Mafia members in the Mafia game. The similarities between a Mafia game and a real network under a botnet attack are briefly illustrated in Figure 1. Several game roles are presented in this figure. However, this paper focuses on the Townies, Mafias, Detective, and Proof. This paper defines new parameters by which the Detective effectively inquests in the game. Moreover, the suggested strategies are used to detect real botnets. The key contributions of this paper are as follows:

- Suggesting efficient detection strategies in detecting the Mafias or malicious nodes.
- Proposing the algorithms that can apply the suggested strategies on the game, using two different techniques.
- Suggesting a new metric based on current common metrics for evaluating the Detective's performance.
- Modeling the problem of botnet detection as a Mafia game to increase the malicious nodes detection rate.

The remainder of this paper is structured as follows. In section II, we propose the detection strategies for the Mafia game and then explain the similarities between a real computer network and the Mafia game. The algorithms for applying the suggested strategy for the Detective on the network is also presented in this section. section III provides the evaluation results of the suggested strategy and its efficiency in a real network. Finally, the conclusion of this paper and our plans for future work are presented in section IV.

II. Proposed Detection Strategies

In this section, we first suggest the detection strategies for the Detective, and then explain how the Mafia game can be mapped on the network to detect the malicious nodes.

The nature of a Mafia game is uncertain, and no specific strategy is always correct in detecting the two sides of the game. However, some of the facts in the game can lead to better detection.

We define five different degrees for each player, based on which, they can be sorted, and the players at the beginning of the sorted list are the suggested candidates for being requested by the Detective. $in(i)$ ($inactive(i)$) is a binary degree, which is 1 if the i^{th} player is not in the game. $ex(i)$ ($exit(i)$) is the number of removed players that were in the Townie team and the i^{th} player has voted them. $pr(i)$ ($proof(i)$) is the number of the i^{th} player's votes for the Proof, only if the Proof roles is revealed. $po(i)$ ($positive(i)$) and $ne(i)$ ($negative(i)$) are binary degrees that are 1 if it is revealed for the Detective in its previous questions that the i^{th} player is in the Townie or the Mafia teams, respectively. Considering these five degrees, the Detective can detect the Mafia team with a higher performance.

There are some cases that the Detective must keep in mind. The players that are previously detected must not be detected again. The $po()$ and $ne()$ degrees specify these cases. Moreover, the winning strategy of the Detective is to detect the players who are probably in the Mafia team. $ex()$ is a good metric for finding the malicious players. The chance that the ones who vote for a removed player from the Townie team are in the Mafia team is high. Furthermore, since the players, who vote for the Proof, are probably Mafias, $pr()$ is another useful metric.

These degrees give each player a score, and the scores are used to sort them based on their priority in being the target of the Detective. Some of these degrees can specify exactly which players must not be requested. For example, a Detective never detects a single player multiple times in this game scenario. It is a waste of chance toward the winning of the Townie team. So, based on the $po()$ and $ne()$ degrees, all the previously detected players are removed from the Detective's target list. However, the exact list and its order are hard to find.

We utilize two techniques to find the probable strategy to lead the Detective inquest of the Mafias. These techniques are linear relation and reinforcement learning. In the linear relation technique, we consider a linear equation of the degrees to calculate the players' final score. For example, we consider that the score of the i^{th} player is $a.ex(i) + b.pr(i)$. Then, we try to find the best coefficients, which are in our example, the values of a and b, based on the Mafia games dataset, that lead

to the best result. On the other hand, in the reinforcement learning technique, we do not consider a linear equation between the final score and the degrees. We train a learning model based on the Mafia games dataset, by which we can find a sorted set of the players as the output. Reinforcement learning is a type of machine learning approach, in which, the problem is passed to an agent, and the agent explores different solutions to find the best one.

The linear technique for detecting the Mafias is shown in Algorithm 1. A nested loop is used for finding the best coefficients of $ex()$ and $pr()$. They are found according to the sorted set of the player's scores. The best result is achieved when the number of Mafia members at the beginning of this set has the highest value. Then, the players are sorted based on the found coefficients, and finally, the inactive players ($in()$), the players who are previously detected ($po()$ and $ne()$), and the Detective itself are removed from the set.

Algorithm 1 The winning strategy of the Detective using linear relation technique.

Require: $\mathcal{G}$, the game parameters available for the Detective
Require: $\mathcal{D}$, the Mafia games dataset
Ensure: Ω, the ordered list of Detective's target candidates

```
in, ex, pr, po, ne ← G
d ← this player
mafias ← the Mafia team based on D
P ← size of in
best ← 0
Ω ← {}
for i ∈ (-20 ... 20) do
    for j ∈ (-20 ... 20) do
        scores ← {}
        for k ∈ (1 ... P) do
            scores ← scores + {k, i.ex[k] + j.pr[k]}
        scores ← sorted scores
        result ← 0
        for k ∈ (1 ... |mafias|) do
            if scores[k][1] ∈ mafias then
                result ← result + 1
        if result > best or Ω = {} then
            best ← result
            Ω ← scores[1]
for k ∈ (1 ... P) do
    if in[k] = 1 or po[k] = 1 or ne[k] = 1 or k = d then
        Ω ← Ω - {k}
return Ω
```

The reinforcement learning technique is presented in Algorithm 2. This technique starts with a training phase based on the number of players at the beginning of the sorted list who is Mafia. When the training phase is over, the players are sorted based on the optimal scores, and then the players who must not be in the output settings are removed.

We can model a network and its security vulnerability against a botnet with the Mafia game, because there are several similarities as follows. The Townies are the legitimate hosts, and the Mafias are the compromised nodes or bots. The interaction between the players is the traffic that is forwarded through the network. The Mafia team cooperates to cause the Townie team to lose their power. The bots in a botnet also cooperate to reduce the power of the hosts in a network. For example, they try to perform a DDoS attack against one of the servers to make the network services unavailable. The players in the Mafia team know each other, while the Townies are not sure about the identity of the other Townies. In a real network, it is the same. The components in a botnet know each other, while in general, the legitimate hosts are not sure which other hosts are trusted. Voting for a player to remove it from the game is similar to sending flooding traffic toward a host to cause it to become unavailable.

Algorithm 2 The winning strategy of the Detective using reinforcement learning technique.

Require: $\mathcal{G}$, the game parameters available for the Detective
Require: $\mathcal{D}$, the Mafia games dataset
Ensure: Ω, the ordered list of Detective's target candidates

```
in, ex, pr, po, ne ← G
d ← this player
mafias ← the Mafia team based on D
P ← size of in
model ← initiate the model
for i ∈ (0 ... 1000) do
    scores ← generate the scores using model
    result ← 0
    for k ∈ (1 ... |mafias|) do
        if scores[k][1] ∈ mafias then
            result ← result + 1
    update model based on result
scores ← generate the sorted scores using model
Ω ← scores[1]
for k ∈ (1 ... P) do
    if in[k] = 1 or po[k] = 1 or ne[k] = 1 or k = d then
        Ω ← Ω - {k}
return Ω
```

The Detective's responsibilities in the Mafia game are similar to that of the intrusion detection component in the network. An intrusion detection component knows malicious patterns and then monitors the traffic from a suspicious node to check its activity patterns [19]. It is similar to the tasks of a Detective in a Mafia game. A honeypot is a deceptive trap in the network that pretends to be a regular host. When the malicious nodes communicate with a honeypot, they are closely monitored and detected [20]. The behavior of the Proof and a honeypot is similar. A honeypot in a network does not have a productive value. So the legitimate host rarely connects with it. The connections with the honeypots are probably established by the illegal nodes (e.g. the bots). The Proof is somehow the same. When the players vote for the Proof, they become suspicious. The malicious nodes fear communicating with the honeypots because this connection leads to their detection. The Mafias are also concerned about voting for the

Proof.

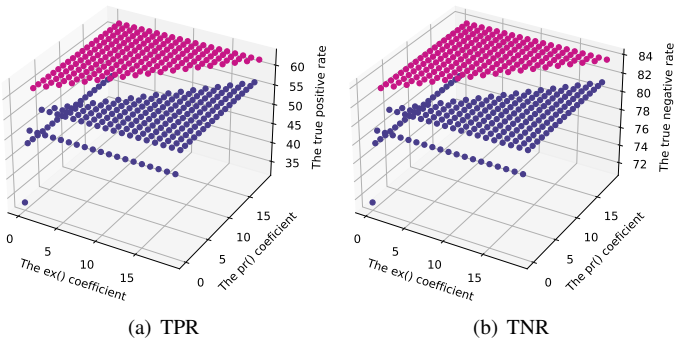
According to these similarities, we can apply the suggested strategies and the proposed techniques (Algorithm 1 and Algorithm 2) of detecting the Mafia players to a real network to detect the malicious nodes. These strategies help the network defender detect the bots or any malicious nodes and then block them. The main goal of modeling a network with a Mafia game is to predict the malicious nodes by their connections. We can say that the connections between the players are more important than the internal activities and detailed features of each node/player.

III. Evaluation

To evaluate the proposed model and the suggested strategy, we should consider two aspects: (1) checking the suggested strategy with the Mafia game datasets and (2) checking the suggested strategy with a real network. The first aspect ensures that the suggested strategy for detecting the Mafia players is effective, and the second one evaluates the performance of modeling a network with the Mafia game. One way to evaluate this effectiveness is to apply the suggested strategies on a real game/network, and then investigate the final result, which is the winning of the Mafia or Townie team. However, we do not access an unbiased group of players, by which we can apply the strategies and check the final winner. Additionally, it is out of the scope of this paper to design an agent that automatically plays the game. Moreover, different factors may affect the final result in a real Mafia game, such as human personality behaviors and players' body language. As a result, we have extended the meaning of two standard metrics in security fields [21], True Positive Rate (TPR) and True Negative Rate (TNR), for each of the strategies, and then evaluate their efficiency using these metrics.

For the first aspect, we need the dataset of real-world Mafia games. As a result, we watched 7 hours of a Mafia game played by Iranian players, which are available on Youtube [22], and then collected their major data to create a Mafia dataset. Initially, ten players are in these games, three of which are in the Mafia team. We compared the evaluation results with the related strategies that are suggested by the previous researches. For the second aspect, we simulated different networks under the Mirai [23] botnet attack.

The two suggested techniques are evaluated in this section. We used a neural network with two hidden layers containing 256 neurons for the reinforcement learning technique. The problem model passed to the agent is a game of sorting different cards, each with specific parameters. For example, in a game with P players, the agent that tries to find the best strategy for the Detective gets P cards, and each card has five values equivalent to the $in()$, $ex()$, $pr()$, $po()$, and $ne()$ degrees. These cards represent each of the players. When the agent sorts the cards, the first card will be suggested to the Detective as its target for inquesting. The action space of the agent contains only two actions. The first action, 0, means the i^{th} card must not be replaced by the j^{th} card. The second action, 1, means that the location of the i^{th} and j^{th} cards must be changed. During the game steps, the values of i and j are changed so that all the cards are considered in the sorting process at the end of the game. 85% of the games in the dataset are used for the training phase, and the remaining 15% are used for testing the learning performance.



(a) TPR (b) TNR

Fig. 2. The value of TPR and TNR for the Detective's strategy using the linear relation technique according to the $ex()$ and $pr()$ coefficients.

In this section, we will discuss the evaluation of the proposed model, including the efficiency of Detectives and network strategies.

A. Evaluating Detective's strategy

The responsibility of the Detective is to detect the Mafia members. In our suggested strategy, the Detective can find them with a higher chance. The defined TPR and TNR values for evaluating the suggested strategy are presented in Equation 1 and Equation 2.

$$\text{TPR}_{CU} = \frac{\text{The no. of targets who are Mafia members}}{\text{The total no. of targets}} \quad (1)$$

$$\text{TNR}_{CU} = \frac{\text{The no. of non-targets who are Townie members}}{\text{The total no. of non-targets}} \quad (2)$$

Based on these metrics [24], we give an example. Consider a game with 3 Mafias and 7 Townies. The sorted list generated by the proposed techniques contains 9 players, because the Detective itself is not in this list. According to the Detective's opinion, the first three members of this set are Mafias and the others are Townies. Hence, if only two of the first three players are Mafias, the values of TPR and TNR are 66% and 83%, respectively.

In Figure 2, the different coefficients of $ex()$ and $pr()$ and their impact on the TPR and TNR values are shown. The best results (i.e. the points with the light color) are obtained where both coefficients are non-zero positive numbers and the coefficient of $pr()$ is greater than or equal to the coefficient of $ex()$. This means that the importance of the $pr()$ degree is higher than the $ex()$ degree for detecting the Mafia players. In other words, the players who have voted for a revealed Proof are more likely to be a Mafia than those who vote for other removed Townies. The greatest achieved values for TPR and TNR are about 61% and 83%, respectively. Another important point about this graph is where the coefficients are zero. In this case, the values of TPR and TNR are low; hence both of the

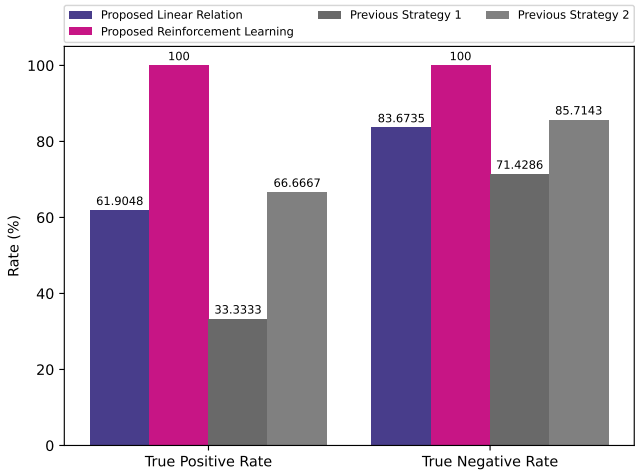


Fig. 3. The values of TPR and TNR by applying the suggested Detective's strategies on the Mafia dataset.

$ex()$ and the $pr()$ degrees must be considered in the winning strategy of the Detective. Since the players who voted one of the Townie members are more likely to be a Mafia, the positive sign of the coefficients is reasonable.

Again, the best-obtained results are reported for comparing the linear relation and reinforcement learning techniques. Hence, to compare this part, we have used the $ex() + 2pr()$ equation as the linear relation. The comparison is illustrated in Figure 3. Based on these results, the reinforcement learning technique can detect 100% of the Mafia and Townie players, using the suggested strategy. On the other hand, the linear relation technique can detect about 61% and 83% of the Mafias and Townies, respectively. We can conclude that the reinforcement learning technique outperforms the linear relation technique when a dataset is available. The other point about these results is that, due to the greater values of TNR against TPR, the suggested strategy works better in detecting the Townies than the Mafias. Although the reinforcement learning technique achieves high TPR and TNR results, we cannot say that the strategy can absolutely detect the Mafias in every game. The results are currently reliable for our used dataset. We also compared the results with the previous related strategies. The proposed reinforcement learning technique has the best results among the other strategies, and the second place belongs to "Previous Strategy 2". On average, the detection rate of our suggested strategy for the Detective is 11% higher than the previous strategies.

B. Evaluating real networks strategy

It is required to evaluate the suggested strategies in real networks [25, 26]. We simulated different networks in Mininet and then propagated the Mirai bots in these networks. Mirai has three main components: the command and control server, the loader, and the bots. The adversary is located on the command and control server and commands the bots to launch a DDoS attack against a specific target. The initial bots scan the network, and whenever they find a host's username and password pair, they send it to the loader. The loader then loads the malicious script on the victim hosts. In the simulated networks, single nodes are dedicated to the IDS, honeypot, command and control, and loader. The number of initial bots and legal hosts varies in different networks. If b is the number of initial bots, the number of legal hosts is $4b + 1$, where b varies from 1 to 10. The network topology with four initial bots is shown in Figure 4. The IDS can detect up to b nodes, and the username and passwords of each host are selected among a set of four pairs.

To calculate the detection efficiency, we use the TPR and TNR values based on the common definition of these two metrics. TPR is a metric for evaluating the power of detecting malicious nodes or bots in the network. This metric can be calculated as in Equation 3.

$$\text{TPR}_{net} = \frac{\text{The no. of detected bots}}{\text{The total no. of detected nodes}} \tag{3}$$

On the other hand, TNR shows the ability to detect the legal nodes in the network. TNR is calculated based on Equation 4.

$$\text{TNR}_{net} = \frac{\text{The no. of undetected legal hosts}}{\text{The total no. of undetected nodes}} \tag{4}$$

Figure 5 shows the TPR and TNR values of applying the suggested strategy of the Detective on the data collected from the simulated networks in Mininet. The results of the linear relation technique show that the detection rate, both for the bots and legal hosts, increases as the number of initial bots gets higher. We can see that when the number of bots are 10, the linear relation technique can detect up to 83% and 95% of the bots and legal hosts, respectively. Moreover, we can see that as the number of initial bots increases, the detection rate of the linear relation technique outperforms the reinforcement learning technique. This shows the power of the linear technique in large-scale networks. Large-scale networks have more connections, and since our strategy is based on the connections, it works better in these networks. On the other hand, we can see that the detection rate of the reinforcement learning technique is 100% when the number of initial bots is one or two. This is because the reinforcement learning agent is trained with the Mafia game dataset, which includes only one Mafia. Therefore, it works better in the cases similar to the Mafia dataset. In general, the average results of both techniques show that the suggested strategy has the TPR and TNR values of about 71% and 91%, which are acceptable detection rates. Furthermore, the gap between the values of TPR and TNR in Figure 5 is similar to the graphs related to Mafia roles strategies evaluations. This similarity is a satisfactory result for claiming that the Mafia game is a good candidate for modeling the security problems of real-world networks.

IV. Conclusion

In this paper, we first proposed some detection strategies for the Mafia game, and then applied them to the computer networks for detecting botnets. Two different techniques, namely linear relation and reinforcement learning, are proposed for implementing the strategies. There is no correlation

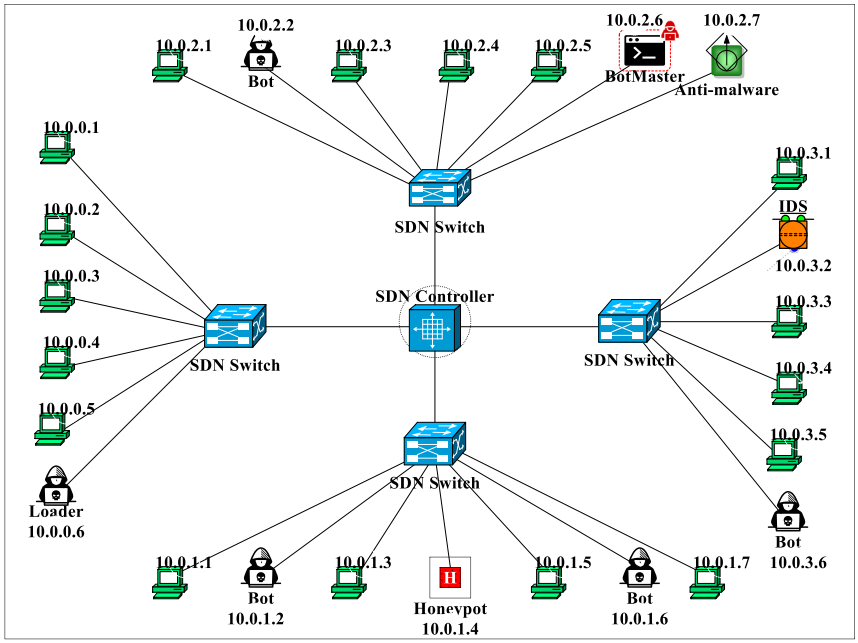


Fig. 4. The simulated network topology with four initial Mirai bots.

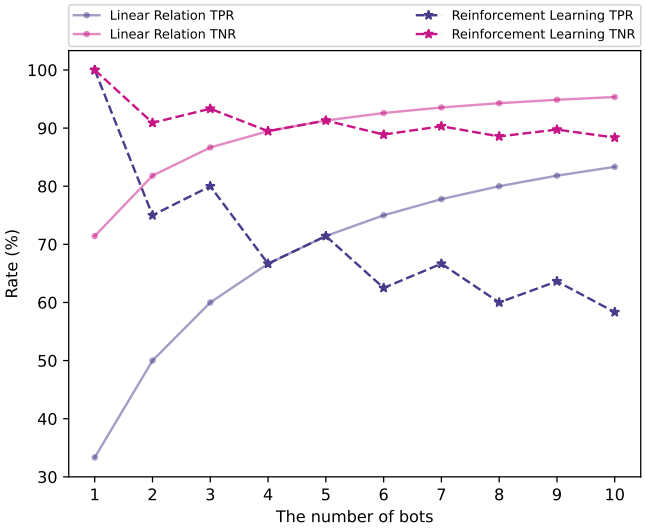


Fig. 5. The values of TPR and TNR by applying the suggested Detective's strategy on the networks simulated in Mininet.

between linearity and reinforcement learning (RL) for the Mafia. They use each concept independently to diagnose bots. We compared two methods to determine which one is more effective. For evaluating them, we extended the TPR and TNR metrics definitions to match the Mafia game. Our suggested strategies are 11% more accurate than the previously suggested strategies in the Mafia game. We also emulated a network that is compromised by the Mirai botnets. We then applied the suggested strategies to check its adaptability with the network. The emulation results obtained from Mininet show that the malicious nodes are detected with a TPR and TNR values of 71% and 91%, respectively, which is a satisfactory result.

We aim to expand our scope in our upcoming research endeavors by incorporating the additional roles outlined in Figure 1 into the Mafia game framework. Our objective is to formulate winning strategies for these newly introduced roles systematically. Additionally, we plan to rigorously assess the performance of a network that has been modeled using our game-based approach, now enriched with these supplementary roles. Looking further into the future, we aim to develop a comprehensive security framework. This framework will encompass many security mechanisms, including but not limited to moving target defense and intrusion prevention systems. Remarkably, our proposed framework will integrate strategies derived from the Mafia game paradigm, effectively leveraging its principles to mitigate various cyber attacks. By bridging the gap between game theory and cybersecurity, our innovative approach holds the potential to create a synergistic alliance between these domains. As we progress, we envision contributing significantly to advancing strategies that not only bolster network security but also offer novel insights into addressing the evolving landscape of cyber threats.

ACKNOWLEDGMENT

This research work is partially supported by the Business Finland 6Bridge 6Core project under Grant No. 8410/31/2022, the Research Council of Finland (former Academy of Finland) IDEA-MILL project (Grant No. 352428), the Research Council of Finland (former Academy of Finland) 6G Flagship program (Grant No. 346208), and the European Union's Horizon Europe research and innovation programme HORIZON-JU-SNS-2022 under the RIGOUROUS project (Grant No. 101095933). The paper reflects only the authors' views. The Commission is not responsible for any use that may be made of the information it contains.

References

[1] P. P. Kundu, T. Truong-Huu, L. Chen, L. Zhou, and S. G. Teo, "Detection and classification of botnet traffic using deep learning with model explanation," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[2] A. Javadpour, F. Ja'fari, T. Taleb, and C. Benzaïd, "Reinforcement learning-based slice isolation against ddos attacks in beyond 5g networks," *IEEE Transactions on Network and Service Management*, 2023.

[3] M. Bagaa, T. Taleb, J. B. Bernabe, and A. Skarmeta, "Qos and resource-aware security orchestration and life cycle management," *IEEE Transactions on Mobile Computing*, vol. 21, no. 8, pp. 2978–2993, 2020.

[4] J. Ashraf, M. Keshk, N. Moustafa, M. Abdel-Basset, H. Khurshid, A. D. Bakhshi, and R. R. Mostafa, "Iotbotids: A novel statistical learning-enabled botnet detection framework for protecting networks of smart cities," *Sustainable Cities and Society*, vol. 72, p. 103041, 2021.

[5] S. I. Popoola, B. Adebisi, R. Ande, M. Hammoudeh, K. Anoh, and A. A. Atayero, "smote-drnn: A deep learning algorithm for botnet detection in the internet-of-things networks," *Sensors*, vol. 21, no. 9, p. 2985, 2021.

[6] F. Ja'fari, S. Mostafavi, K. Mizanian, and E. Jafari, "An intelligent botnet blocking approach in software defined networks using honeypots," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 2, pp. 2993–3016, 2021.

[7] C. Joshi, R. K. Ranjan, and V. Bharti, "A fuzzy logic based feature engineering approach for botnet detection using ann," *Journal of King Saud University-Computer and Information Sciences*, 2021.

[8] R. Abu Khurma, I. Almomani, and I. Aljarah, "Iot botnet detection using salp swarm and ant lion hybrid optimization model," *Symmetry*, vol. 13, no. 8, p. 1377, 2021.

[9] R. Musotto and D. S. Wall, "More amazon than mafia: analysing a ddos stresser service as organised cybercrime," *Trends in Organized Crime*, pp. 1–19, 2020.

[10] A. Javadpour, F. Ja'fari, T. Taleb, M. Shojafar, and B. Yang, "Scema: an sdn-oriented cost-effective edge-based mtd approach," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 667–682, 2022.

[11] A. Javadpour, F. Ja'fari, T. Taleb, and M. Shojafar, "A cost-effective mtd approach for ddos attacks in software-defined networks," in *in Proc. IEEE GLOBECOM'22, Rio De Janeiro, Brazil*, Dec. 2022.

[12] C. Benzaïd, M. Boukhalfa, and T. Taleb, "Robust self-protection against application-layer (d) dos attacks in sdn environment," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2020, pp. 1–6.

[13] MafiaScum, "Mafiascum wiki," https://wiki.mafiascum.net/, 2021, [Accessed: June 2022].

[14] Q. Chang, "A simulator for analyzing the balance of mafia game," in *2020 IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS)*. IEEE, 2020, pp. 622–627.

[15] M. Kondoh, K. Matsumoto, and N. Mori, "Development of agent predicting werewolf with deep learning," in *International Symposium on Distributed Computing and Artificial Intelligence*. Springer, 2018, pp. 18–26.

[16] M. Hagiwara, A. Moustafa, and T. Ito, "Using q-learning and estimation of role in werewolf game," in *Proceedings of the Annual Conference of JSAI 33rd (2019)*. The Japanese Society for Artificial Intelligence, 2019, pp. 2O5E303–2O5E303.

[17] X. Bi and T. Tanaka, "Human-side strategies in the werewolf game against the stealth werewolf strategy," in *International Conference on Computers and Games*. Springer, 2016, pp. 93–102.

[18] T. Wang and T. Kaneko, "Application of deep reinforcement learning in werewolf game agents," in *2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. IEEE, 2018, pp. 28–33.

[19] A. K. Sangaiah, A. Javadpour, F. Ja'fari, P. Pinto, W. Zhang, and S. Balasubramanian, "A hybrid heuristics artificial intelligence feature selection for intrusion detection classifiers in cloud of things," *Cluster Computing*, pp. 1–14, 2022.

[20] H. Wang, H. He, W. Zhang, W. Liu, P. Liu, and A. Javadpour, "Using honeypots to model botnet attacks on the internet of medical things," *Computers and Electrical Engineering*, vol. 102, p. 108212, 2022.

[21] K. Sinha, A. Viswanathan, and J. Bunn, "Tracking temporal evolution of network activity for botnet detection," *arXiv preprint arXiv:1908.03443*, 2019.

[22] M. Plus, "Mafia plus," https://www.youtube.com/channel/UCzZJv5EsGb-h1HDEPa2o4Bg/videos, 2020, [Accessed: June 2022].

[23] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *26th USENIX security symposium (USENIX Security 17)*, 2017, pp. 1093–1110.

[24] A. Javadpour, S. K. Abharian, and G. Wang, "Feature selection and intrusion detection in cloud environment based on machine learning algorithms," in *2017 IEEE international symposium on parallel and distributed processing with applications and 2017 IEEE international conference on ubiquitous computing and communications (ISPA/IUCC)*. IEEE, 2017, pp. 1417–1421.

[25] M. L. Adjou, C. Benzaïd, and T. Taleb, "Topotrust: A blockchain-based trustless and secure topology discovery in sdns," in *2022 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 2022, pp. 1107–1112.

[26] Z. Shu, H. Feng, T. Taleb, and Z. Zhang, "A novel combinatorial multi-armed bandit game to identify online the changing top-k flows in software-defined networks," *Computer Networks*, vol. 230, p. 109783, 2023.