Strategic Defense against Stealthy Link Flooding Attacks: A Signaling Game Approach

Abdullah Aydeger, *Member, IEEE*, Mohammad Hossein Manshaei, *Member, IEEE*, Mohammad Ashiqur Rahman, *Senior Member, IEEE*, Kemal Akkaya, *Senior Member, IEEE*

Abstract—With the increasing diversity of Distributed Denial-of-Service (DDoS) attacks, it is becoming extremely challenging to design a fully protected network. For instance, Stealthy Link Flooding Attack (SLFA) is a variant of DDoS attacks that strives to block access to a target area by flooding a small set of links, and it is shown that it can bypass traditional DDoS defense mechanisms. One potential solution to tackle such SLFAs is to apply Moving Target Defense (MTD) techniques in which network settings are dynamically changed to confuse/deceive attackers, thus making it highly expensive to launch a successful attack. However, since MTD comes with some overhead to the network, to find the best strategy (i.e., when and/or to what extent) of applying it has been a major challenge. The strategy is significantly influenced by the attacker's behavior that is often difficult to guess. In this work, we address the challenge of obtaining the optimal MTD strategy that effectively mitigates SLFAs while incurs a minimal overhead. We design the problem as a signaling game considering the network defender and the attacker as players. A belief function is established throughout the engagement of the attacker and the defender during this SLFA campaign, which is utilized to pick the best response/action for each player. We analyze the game model and derive a defense mechanism based on the equilibria of the game. We evaluate the technique on a Mininet-based network environment where an attacker is performing SLFAs and a defender applies MTD based on equilibria of the game. The results show that our signaling game-based dynamic defense mechanism can provide a similar level of protection against SLFAs like the extensive MTD solution, however, causing a significantly reduced overhead.

Index Terms—Stealthy Link Flooding Attack, Crossfire Attack, Signaling Game, Moving Target Defense

1 Introduction

A recent report by Netscout states that Distributed Denial-of-Service (DDoS) attacks will continue to grow [1]. According to the report, last year 1.35 Terabits per second (Tbps) DDoS traffic hit Github, and just after five days of that incident, 1.7 Tbps DDoS traffic launched against an unnamed US-based service provider. Even though there are many DDoS defense mechanisms available [2], they are not capable of competing with some recent types of attacks. For instance, there is a new kind of attacks where an adversary does not attack the target network/server directly like in traditional Link Flooding Attacks (LFAs) [3], [4]. In this recent kind, an adversary utilizes a high number of bots in distributed locations to sends a small number of packets to a set of servers around the target network/server such that the communication links to access the target becomes congested. This type of attacks are known as Stealthy LFAs (SLFAs). The Crossfire attack is an exemplary kind of SLFAs, and it is primarily performed by congesting the communication links surrounding the network of the target servers by sending low-volume traffic over them from many bots in distributed locations. Since the traffic the bots send is legitimate, and they do not attack the servers directly, it is very challenging to detect such attacks using traditional mechanisms. The consequence of this attack is the blockage of external access to the servers while they are still active/running without any problem within the network.

The concept of Moving Target Defense (MTD), in which the defense is done dynamically, often proactively, by introducing agility to the network behavior, is proven useful to defend against such stealthy attacks [5], [6] [7], [8]. This agility brings protection to the system by providing resistance, as it complicates the tasks of an attacker by adding inconsistency or confusion in the knowledge of the system. These features can be implemented in various ways including but not limited to changing IP addresses of network devices, the operating system of servers, and routing information, more often by leveraging the capabilities of Software Defined Networking (SDN). MTD, more specifically Random Route Mutation (RRM) [9], is found useful in defending a network system against SLFAs [8]. However, RRM brings significant overhead to the network since it takes time to update the system characteristic (i.e., routing information), and this process is usually applied not only to malicious users but also to legitimate clients. This is because the defender is not aware of the type of a client. Therefore, the concern of when and for whom to change system parameters in order to minimize the cost of the defender and impact of the attacker becomes a critical issue and it should be investigated thoroughly.

Even though there are many research works proposing defense mechanisms against SLFAs in the literature, they are

- *Abdullah Aydeger is with the Department of Computer Science, Southern Illinois University, Carbondale, IL, 62901 (aydeger@cs.siu.edu).*
- *Mohammad Hossein Manshaei, Mohammad Ashiqur Rahman, and Kemal Akkaya are with the Department of Electrical and Computer Engineering, Florida International University, Miami, FL 33174, USA (e-mails: mmanshae, marahman, kakkaya}@fiu.edu).*
- *M. H. Manshaei is also with the Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran.*
- *A. Aydeger and M. H. Manshaei are the co-first authors.*
- *A. Aydeger is the corresponding author.*

mostly reactive solutions in which the attacker made some harm before the attack is mitigated [10], [11]. In addition to these works, there are other kinds of proactive defense mechanisms that apply detection of bots, sending forged responses to the hosts, or similar approaches [12], [13]. We argue that bot detection might not be possible considering that bots often behave similarly to legitimate hosts, and serving fake replies to hosts is not acceptable since that may cause letting the legitimate clients use the longer paths and having more delays. Furthermore, entirely disabling the network reconnaissance offers a solution against these attacks since the attacker will not be able to gather the necessary information to organize an attack. However, we argue that blocking all the network reconnaissance packages would prohibit legitimate usage of the network debugging techniques, which is the main usage of these packets. Notably, these packets can be utilized by a network tester, a person who may need access to these network diagnosis tools by sending network reconnaissance packets while s/he may not have full privilege of SDN Controller's entire view of the network. These surveillance activities are used not only for routing information, which could be easily obtained from the SDN Controller but also for error detection on the network routers/links and Quality of Service related issues such as round trip time (RTT) computation of these packets and available bandwidth by standard reconnaissance packets [14]. The usage of these tools also provides validation of compliance and correctness of network devices/environments [15]. Hence, it is claimed that entirely disabling these packets would cause other problems related to network diagnosis [16]. As another potential solution, applying MTD approaches (i.e., RRM) would lessen the critical link load and mitigate SLFA [8]. However, the MTD actions, e.g., frequent route mutations, especially when the system is not under attacks, would bring unnecessary costs to the network in terms of additional delay and increased packet losses. Therefore, there is a need for a proactive and dynamic solution that can protect the network system from the SLFA while bringing minimal service degradation to the hosts. The major challenges that we have addressed in this work to provide this solution are as follows: (1) to design a real-time defense system against SLFA, (2) to select/decide dynamically whom to deceive, and (3) to make optimal decision, i.e., how often should the mitigation of routes against specific host be applied.

In order to model such a system that achieves the best efficiency of RRM, we need to design an intelligent defense mechanism that can take into account the above concerns for running RRM. To design such a strategic defense mechanism, in this paper, we apply a signaling game [17] to model and analyze the attack and defense actions together. The results from this analysis assist in applying RRM selectively and appropriately such that the target network can remain sufficiently protected against Crossfire attacks and the legitimate clients experience a minimal cost. In the signaling game, a player constructs a belief about the type of opponent. This belief is always updated by the opponent's actions. Then, the player can better decide about its optimal strategy, given its belief about the type of player from which it receives messages. We can consider the following scenario as a real-world example of how the attack and defense evolves. In order to complete a successful attack, an adversary needs to run a few steps. For example, in our case, when an adversary intends to run the Crossfire attack, several steps must be taken for a successful output. The first step is the scanning/reconnaissance, where the attacker looks for the links to attack. This reconnaissance step continues until the necessary attack points are not revealed; hence we need to follow these steps to countermeasure the final attack. Our defined game must be played in a real-time setting to deceive the attacker in the first step. Essentially, we consider the attacker and the defender as the players of this signaling game. We analyze the game and compute all Bayesian Nash equilibria. The game results are then used to decide when and for whom to perform RRM. We name the corresponding defense mechanism as *Strategic RRM*.

To evaluate our proposed solution, we first implement a network using Mininet, a virtual SDN testbed [18], and the defense mechanism on FloodLight Controller [19]. Then, we run experiments with extensive RRM, in which the routing paths are periodically changed, to observe the overhead to the legitimate clients when there is not any attack in place. Next, we run the SLFA and realize the reasonable frequency for periodic RRM, which is used later to compare with *Strategic RRM*. Moreover, we consider two kinds of attackers based on attack approach and capability. In each case, we report the number of packet losses when the defender has either no protection, periodic RRM or *Strategic RRM*. We show that while periodic RRM provides a significant improvement to the network defense, it introduces packet delay as well as packet losses even when there is no attacker in the system. We also show that *Strategic RRM* offers a similar defense performance as periodic RRM while it causes much less overhead.

In summary, our contribution to addressing the challenges given in this paper is threefold: First, we model the attacker and the defender as players of a signaling game and define their actions and payoff models. Second, we design a belief function that considers the characteristics of the attack model. Third, we solve the game and derive all possible Nash equilibria of this game from obtaining the best strategies for the defender and the attacker. According to the game results, we then design a strategic defense mechanism to mitigate SLFAs. Finally, we evaluate the proposed mechanism by conducting extensive experiments considering different attack approaches and capabilities for different defense strategies.

This paper is organized as follows. In Section 2, we discuss the preliminaries and relevant work. In Section 3, we present the game model. In the following section, we analyze the game and design a defense mechanism. Detailed performance evaluation of the proposed work is presented in Section 5. Finally, we conclude the paper in Section 7.

2 Preliminaries

In this section, we briefly discuss some concepts that we applied in this research.

2.1 Moving Target Defense

A system typically offers resources for its clients' usage. A client accesses these resources through interfaces. An interface is often considered as an access points to the system.

These access points create different attack surfaces with various levels of capabilities and impacts on the system [20]. The configurations of these attack surfaces usually do not change over time. The static nature of attack surfaces helps launch attacks easily. Once an attacker gains the system information (IP addresses, Operating System, routing, etc.), s/he can use that knowledge to exploit the system vulnerabilities.

MTD is a defense mechanism that provides security through adding agility/dynamicity to the system, and thus by changing the attack surface [21]. The system's dynamicity deceives the attacker and increases his/her effort to launch a successful attack [8]. There are several methods to perform MTD, including updating software or network characteristics of the system. Route mutation-based MTD has been shown useful in defending against SLFAs [7].

2.2 Software Defined Networking

SDN proposes the separation of routing decisions from forwarding tasks [22]. An SDN-based network architecture typically considers three layers: application, control, and data. While the data layer (i.e., network switch) is responsible for forwarding the data from one port to another, the control layer (i.e., SDN Controller) is responsible for making the decisions on routing rules. The application layer contains network functions including but not limited to Intrusion Detection System (IDS), firewall. It provides great flexibility and efficiency for network management [23]. Thus, we leverage SDN to implement our solution.

2.3 Attack Model

Stealthy link flooding attacks, such as the Coremelt [3] and the Crossfire [4], have drawn lots of attention from researchers recently. SLFAs are indirect attacks such that the attacker targets the communication links instead of targeting the server directly [24].

The attack model of this research considers Crossfire attacks as the threat and the Autonomous System (AS) as the target domain. Fig. 1 shows a Crossfire attack scenario. As it can be seen from the Fig. 1, there is a target area that the attacker plans to congest using the bots. The attack consists of two phases: reconnaissance and attack. In the first phase, the attacker uses bots residing at distributed locations to obtain how network communication links and routes are formed toward the target area. The attacker uses traceroute packets to obtain the information about the network topology and traffic routes. Leveraging this information, s/he obtains the critical communication links that carry most of the data to the target area. Finding these links is important as they will be used in the second phase of the attack. The attacker should also have the information about other servers in the network. Some of these servers are used to send traffic from bots in order to occupy the communication links. These servers are called decoy servers. While the bots, often many in number, send regular traffic to the decoy servers to evade Intrusion Detection Systems (IDSs), the critical links suffer with congestion, causing the target area isolated from its clients.

Two different approaches are considered for an attacker:

- *Stealthy Attacker*: In this model, the attacker is not in a rush to conduct the attack. S/he runs reconnaissance, and accordingly creates a target link set. Sometimes s/he stays idle in order to behave like legitimate users. After that, s/he starts the attack phase to exhaust the links' bandwidth.
- *Aggressive Attacker*: In this approach, the attacker behaves more greedy and transmits more quickly to congest the communication links. Reconnaissance attack is performed, the information is collected and the attack pattern is designed within a given short time for the aggressive attacker.

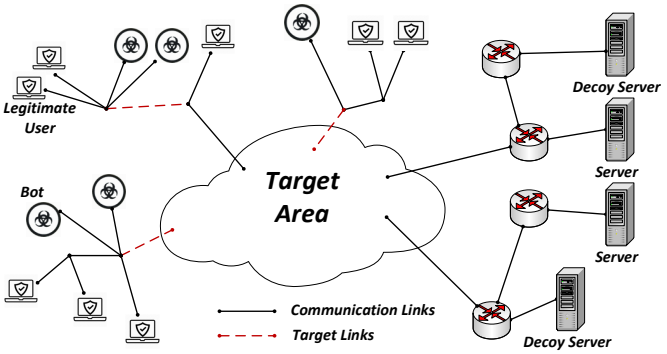


Fig. 1: The Crossfire Attack

Furthermore, an attacker often has a limited number of bots. Thus, two qualitative capabilities are considered:

- *Decent Attacker*: The attacker has a small number of bots to launch a Crossfire attack.
- *Strong Attacker*: The attacker has an extended capability (i.e., a significantly increased number of bots) compared to a decent attacker.

2.4 Proposed Defense Mechanism

Our proposed defense mechanism is based on a *signaling game*. Note that the signaling game is a dynamic game, in which the first move is made by the bot or legitimate user, and the defender is the second player. The game is incomplete information because the defender does not know with whom it is playing. This uncertainty is modeled with Nature. Moreover, we update the knowledge of the defender in repeated interactions with other nodes. Both players decide their actions according to their expected outcomes. We explain the details of the game model in Section 3. The high-level architecture of our defense procedure is shown in Fig. 2. As can be seen in Fig. 2, our system first initializes the parameters depending on the network environment and topology. The initial parameters are calculated using preliminary experiments (no attack scenario). In the real-world, these parameter values can be realized by observing network delay increase or throughput change in case of route update. Then, the defense mechanism checks if the conditions to trigger the defense strategy, is satisfied. The trigger condition is related to the game solution and later discussed in Section 4. If the decision is to defend, RRM is employed, and system parameters are updated. If not, the defender stays idle until it receives an action from the attacker. Each time an action is received from the attacker, the defender updates his/her parameters and checks if it is time to defend.

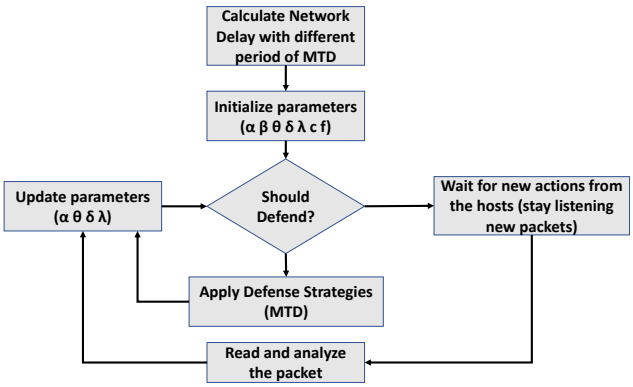


Fig. 2: The procedure of defense decision

TABLE 1: Symbols used in manuscript

Symbol	Definition
N	Reconnaissance packet sender
G	Regular packet sender
$\overline{R}$	Defender's action as no RRM
R	Defender's action as RRM
α	Attacker's gain from single reconnaissance packet
θ	Belief value of whether the client is legitimate or bot
β	Cost for renting bots
δ	Cost of not providing information to legitimate user
λ	Gain of defending through reconnaissance
c	Cost of applying RRM
f	Frequency of Random Route Mutation (RRM)
N_c	Number of hosts in the network

3 Crossfire Attack and Defense Game Model

In this section, we first give an overview of the proposed game. Then, we define the players' action sets and model the belief function and payoffs.

3.1 Overview

In our Crossfire Attack and Defense Game model, a client can be malicious (a bot corresponding to a Crossfire attacker) or benign. Clients send various requests to the servers in the target network. A bot makes ping and traceroute requests so that the attacker can do the reconnaissance (i.e., link map construction) to launch a Crossfire attack. The security administrator of the target network (simply the defender) can apply RRM to thwart this reconnaissance, so the data phase of the Crossfire attack. In the case of applying RRM, the defender randomly changes/mutates the routing paths, thus changing the link map. We model the interaction between the Crossfire attacker (in fact, each bot individually) and the defender using a *signaling game* $\mathcal{G}^{cf}$. Signaling game is a two-player incomplete information game, in which Nature has a unique randomizing strategy (i.e., θ) that is commonly known to both the defender and the attacker. With this randomizing strategy the type of sender would be defined.

The first player, a.k.a. sender (here, the attacker/bot), is informed of Nature's choice and chooses an action. The second player, a.k.a. the receiver (here, the defender), then chooses an action without knowing Nature's choice but observing the first player's action. Our choice of the signaling game is based on the dynamic and incomplete information characteristic of the Crossfire attack, where the action of one player is conditioned over its belief about the type of the opponent. The action flow of the game is shown in Fig. 3.

The game $\mathcal{G}^{cf}$ is played individually with each client/sender who can be a bot or a legitimate user. A bot is considered as a type t_1 sender, while a legitimate user as a type t_2 sender. We represent this set of sender types as $\mathbb{T}$, i.e., $\mathbb{T} = \{t_1, t_2\}$. The second player of game $\mathcal{G}^{cf}$ is the defender. The game is played in the following steps, as shown in Fig. 3. The defender receives two different types of messages, i.e., *Reconnaissance* or *Regular Traffic*. Nature draws type t_1 or t_2 with a probability of θ or $1-\theta$, respectively. The defender does not know exactly whether the observed message is coming from a bot or a legitimate user but s/he can only form/follow a belief. According to this belief, the defender must decide whether to defend by applying RRM or not. In the following subsections, we define the actions of the players, and accordingly model the belief and payoff functions. In the game, we often use the term "attacker" to represent a bot.

3.2 Action Sets

In game $\mathcal{G}^{cf}$, we define the action set $\mathbb{A}$ of the first player by an ordered pair $(m(t_1), m(t_2))$, where $m(t_1)$ is the action of type t_1 (i.e., a bot) and $m(t_2)$ is the action of type t_2 (i.e., a legitimate user). We also assume that each sender, irrespective of his/her type, can select his/her action from the same action set. Let $\mathbb{A}$ be $\{N, G\}$, where N represents sending reconnaissance packets while G is about sending the usual/regular traffic. It is worth noting that if the attacker sends reconnaissance packets (i.e., plays N), then the defender may become suspicious. On the other hand, when the attacker sends regular traffic (i.e., plays G), s/he will not gain necessary information.

By using these actions, the attacker can make a strategic plan with a combination of actions. This strategy could either be slow and stealthy, or quick and greedy. If the attacker's strategy is stealthy one as defined in Section 2.3, a longer time will required to do the reconnaissance. In this long period, there is a higher possibility to have changes in the link map due to existing defense strategies (or even usual network events). Hence, the attack will have less chance to be successful. On the other hand, if the attacker is so greedy about the attack, s/he might immediately do harm yet possibly end up being caught quickly. Thus, the attacker needs to do a trade-off between the time spent for reconnaissance and the risk of being detected by the defender. We will elaborate this issue in more detail in Section 5.

Similar to the sender, the receiver/defender has an action set $\mathbb{B}$. It can respond/reply with a route mutation to deceive the attacker (i.e., play R). Otherwise, it reply with no route change (i.e., play $\overline{R}$). Hence, $\mathbb{B} = \{R, \overline{R}\}$. If the defender applies route mutation (i.e., R), then there will be an extra cost depending on the time it takes to install flow rules on network devices. The cost for the regular replies (i.e., $\overline{R}$) is considered as zero, since there is not any additional efforts required. We will later elaborate on the defense cost when we obtain the defender payoffs.

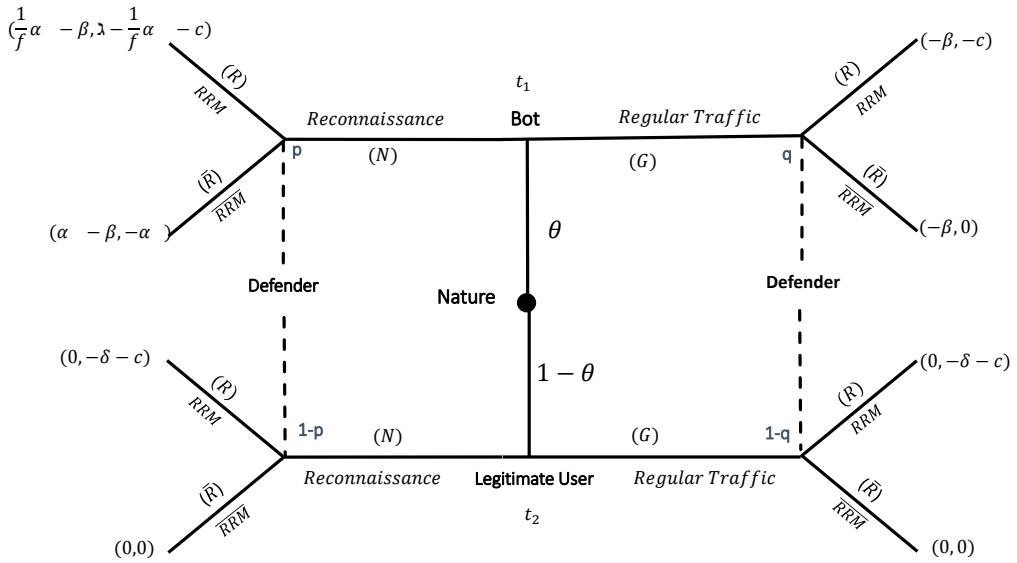


Fig. 3: $\mathcal{G}^{cf}$ signaling game model representation.

3.3 Belief Model

In game $\mathcal{G}^{cf}$, the defender does not know whether it receives messages from an attacker or a legitimate user. Yet, it observes these messages and builds the belief function (i.e., θ) about the sender's type based on this observation. Remember that this θ is a common knowledge between the sender and the receiver. The defender has a different belief for each client/sender: θ_y, where $0 < y \leq N_c$. Furthermore, to represent the belief at a time instance x, we use the notation $\theta_y(x)$. The belief ranges from 0 to 1, while 0 showing no sign of suspicion of the client's legitimacy. The belief has two parts: the initial belief that is statically calculated at the beginning of the game and the dynamic belief that is updated according to the sender's actions.

3.3.1 Initial Belief

For experimental purposes of our work, the belief function for each node (i.e., $\theta_y(0)$) starts with zero since we do not consider/assume any prior information (ground truth) to suspect the node. However, in reality, the ground truth can be assigned to some nonzero value based on some criteria if there is any prior information (i.e., IP addresses based on some countries are more suspicious than others).

3.3.2 Dynamic Belief

The dynamic belief for a specific sender will be calculated each time the game is played. The belief is defined based on three different weighted factors, as shown in Equation (1). While these factors are normalized values, the summation of their weights are 1 (100%). The first factor is the previous belief (i.e., $\theta_y(x-1)$) that is weighted by F_1. The reason is that if a client has been behaving suspicious or benign for most of the time, his/her current action, which may be different than this characteristic, cannot eliminate (or changes) his/her performance previously observed/learned. That is why, we consider assigning a high weight (e.g., 90%) to F_1.

$$\theta_y(x) = \theta_y(x-1) \times F_1 + \frac{\sum_{k=1}^{N_c} \theta_k(x-1)}{N_c} \times F_2 + A(x) \times F_3 \quad (1)$$

The second factor is the average of the beliefs for all the network hosts (i.e., $\frac{\sum_{k=1}^{Nc} \theta_k(x-1)}{Nc}$). The network hosts usually are independent of each other. However, in the reconnaissance step, the attacker's goal is to map out the critical links in which many bots are often deployed to succeed. Therefore, some network hosts (e.g., bots) are dependent on each other and managed by the attacker (e.g., the botmaster) in such a way that they act together to collect current reconnaissance responses of the servers. Each host's belief value is affected by other hosts in the network to emphasize this vital bot collaboration. Thus, we consider this collective behavior metric as a crucial part (e.g., 9%) of our belief equation and represent its weight by F_2.

The last metric we use to update our belief is the current/latest packet that we have received from the sender (i.e., $A(x)$). The value of $A(x)$ is equal to either 1 if the packet received is a reconnaissance packet, and 0 if the packet is a data packet. It's weight is F_3, which we consider as small (e.g., 1% Even though its weight is considered small, it's impact on the belief adds up rapidly as the sender keeps sending packets to the receiver, and the game is continuously played for each packet receipt.

3.4 Payoff Model

In this section, we calculate the benefit and cost of each of the players considering all possible strategies. Let us first consider the case where the game is played between the attacker/bot (i.e., the sender type is t_1) and the defender. This case is represented by the upper part of the signaling game as shown in Fig. 3.

If the defender applies RRM (i.e., plays R) in response to a reconnaissance message (i.e., the sender plays N) from the bot (i.e., given that Nature has chosen the bot to play with a probability of θ), the defender can successfully defend the targeted server(s) against the attack. Let us assume that α represents the number of packet losses that the attacker can cause by leveraging the information that is collected through the reconnaissance process. In other words, α is

the numerical gain of the attacker. If we assume that the frequency of running RRM, is f, then the benefit of the attacker will be degraded for the higher frequency of RRM. Hence,in this case, we model the benefit of the attacker by $\frac{1}{f}\alpha$. We designate the attacker's cost by β, which is an increasing function on the number of bots that are deployed by the attacker in the network. This parameter also represents the capability of the attacker in our experiments: a decent or strong capability. While the decent attacker can afford a small number of bots, the strong attacker can deploy more bots. The number of packets sent by the attacker can be considered as another metric of the attacker's cost, yet this does not bring a direct cost to the attacker. Finally, considering the calculated benefit and cost for the attacker we conclude that the attacker's payoff is $\frac{1}{f}\alpha - \beta$.

We represent the benefit of the defender (gain of defending against reconnaissance) by λ. This benefit is made by giving the attacker wrong information about reconnaissance, which prevents the data phase of the attack. This action brings protection to the network in terms of decreased packet loss. Moreover, we represent the total cost of applying RRM by c, which is measured in terms of additional delay and packet loss that might be caused due to the flow table update. Given the calculated cost and benefit for the defender, the payoff of the defender running RRM against the bot is: $\lambda - \frac{1}{f}\alpha - c$.

Following the above discussion for the payoffs of the attacker and defender, if the defender does not react to the reconnaissance messages of the bot, the payoff of the attacker and defender will be $-\alpha$ and $\alpha - \beta$, respectively. With a similar analysis, we can show that the payoff of the defender is $-c$. If it does not defend, the payoff would be 0. The payoff of the attacker when s/he does not send packets will always be equal to $-\beta$.

Now we consider the second game, where the players are the legitimate user and the defender (i.e., the lower part of the signaling game presented in Fig. 3). Since RRM does not have significant impact on the legitimate users' traffic, we consider its payoff as zero for all possible actions of the defender. However, running RRM against legitimate users generates wrong information to the legitimate user, and it may cause a problem with troubleshooting. We model this effect by parameter δ. Hence, the payoff of the defender when it runs RRM will be equal to $-\delta - c$.

4 Game Analysis and Protocol Design

In the following, we first examine game $\mathcal{G}^{cf}$ for the existence and properties of pure strategy Perfect Bayesian Nash Equilibria (PBNE). We then use our analysis to design a defensive protocol to optimize defender strategies against Crossfire attacks.

4.1 Game Analysis: Perfect Bayesian Nash Equilibrium

In complete information or non-Bayesian games, a strategy profile is a Nash equilibrium (NE) if every strategy in that profile is a best response to every other strategy. However, players in Bayesian games would like to maximize their expected payoffs, given their beliefs about the other players [25]. A PBNE is characterized as a strategy profile and belief that satisfy the following four requirements [26]:

Requirement 1: After observing any message m_j from sender j, the defender must have a belief about which types could have sent m_j. Denote this belief by the probability distribution $\mu(t_i|m_j)$, where $\mu(t_i|m_j) \geq 0$ for each type t_i, and $\sum_{t_i \in T} \mu(t_i|m_j) = 1$

Requirement 2: For each message m_j, the defender's action $a^*(m_j)$ must maximize his expected utility u_d, given the belief $\mu(t_i|m_j)$ about which type could have sent m_j. That is, $a^*(m_j)$ satisfies:

$$\max_{m_j \in M} \sum_{t_i \in T} \mu(t_i|m_j) u_d(t_i, m_j, a(m_j))$$

Requirement 3: For each type t_i, the sender's (whether a bot or a legitimate user) message $m^*(t_i)$ must maximize his utility (u_d), given the defender's strategy $a^*(m_j)$. That is, $m^*(t_i)$ satisfies:

$$\max_{m_j \in M} u_d(t_i, m_j, a^*(m_j))$$

Requirement 4: For each $m_j \in M$, if there exists type t_i such that $m^*(t_i) = m_j$, then the defender's belief at the information set corresponding to m_j must follow from Bayes' rule and the sender's strategy:

$$\mu(t_i|m_j) = \frac{p(t_i)}{\sum_{t_i \in T_j} p(t_i)}$$

where T_j denotes the set of types that send the message m_j. Considering the above requirements, we can now define the PBNE.

Definition 1. *A pure-strategy PBNE in a signaling game is a pair of strategy $m^*(t_i)$ and $a^*(m_j)$ and a belief $\mu(t_i|m_j)$ satisfying Requirements 1 to 4.*

In the following, we use $(p, 1-p)$ and $(q, 1-q)$ to denote the second player's (the defender) beliefs at its two information sets. Recall that for the defined signaling game in Figure 3, the sender's pure strategy determined by an ordered pair $(m(t_1), m(t_2))$ where $m(t_1)$ and $m(t_2)$ are the chosen strategies by user types t_1 and t_2, respectively. Note that in our model t_1 and t_2 are bot and legitimate types. Similarly, the defender's strategy is determined by an ordered pair $(a(N), a(G))$, in which $a(N)$ and $a(G)$ demonstrate the defender strategy following the sender's reconnaissance and regular traffic signals, respectively.

Furthermore, a pure strategy PBNE profile is determined as tuple $\{\mathcal{S}_1, \mathcal{S}_2, p, q\}$, in which $\mathcal{S}_1$ is the pair of the sender strategy chosen by each type (whether bot or legitimate user), $\mathcal{S}_2$ is the pair of defender strategy in response to each signal, and p and q are attacker belief concerning the type of sender for reconnaissance (N) or regular (G) signal, respectively. According to the sender pure strategy, two kinds of PBNE could exist in signaling game, called *pooling* and *separating*.

A PBNE is called *pooling equilibrium* if $m(t_1) = m(t_2)$. In other words, the bot and legitimate user send the same signal, regardless of their types. In contrast, a PBNE is called *separating equilibrium* if $m(t_1) \neq m(t_2)$, i.e., the bot and legitimate users send a different signal, depending on their types. We now examine $\mathcal{G}^{cf}$ for (pure) PBNE. We first probe the existence of pooling equilibria.

Given the definition of pooling and separating equilibrium, in the following, we derive all conditions under which there exist these Nash equilibrium profiles in our defined game. In other words, since the values of payoffs vary given the topology of the network and the period of sending different packets and probes in the reconnaissance phase, there would be different conditions to be checked for the existence of these Nash equilibrium points.

Theorem 1. *For any values of θ, there exists a pooling equilibrium on N, in $\mathcal{G}^{cf}$ signaling game.*

Proof. Let us suppose that there exists a pooling NE with (N, N) strategy for the sender. Then the defender's information set corresponding to N is on the equilibrium path, so the defender's beliefs $(p, 1-p)$ at this information set is determined by Bayes' rule and sender's strategy: $p = \theta$. We first compute the expected payoff of the defender given its belief. The defender's expected payoff for playing R is:

$$\theta \times (\lambda - \frac{1}{f}\alpha - c) + (1-\theta) \times (-\delta - c) \tag{2}$$

And defender's expected payoff for playing $\overline{R}$ is:

$$\theta \times (-\alpha) + (1-\theta) \times (0) \tag{3}$$

Comparing the above payoffs for the defender we can define a threshold for belief, called θ^*, which is equal to $\frac{\delta+c}{\delta+\alpha(1+\lambda-\frac{1}{f})}$. We first assume that $\theta^* \leq 0$, then the following two cases could take place for the dominant strategy of the defender:

- $\theta \geq \theta^* := \frac{\delta+c}{\delta+\alpha(1+\lambda-\frac{1}{f})}$: Therefore playing R dominates $\overline{R}$, following N signal, which can be easily verified by comparing the expected payoffs presented in Equations (2) and (3).
  Now we should check whether the senders have incentives to deviate from the N strategies, given the defender strategy which is R in this case. If the defender chooses strategy R for responding to message G, there is no incentive for the senders to deviate from their strategies. In fact, in that case, the sender of type 1 (t_1 or Bot) achieves $-\beta$ instead of $\frac{1}{f}\alpha - \beta$. The sender of type 2 ($t_2$ or legitimate user) obtains 0 in both cases. Hence, there is no incentive for the senders to deviate from strategy N.
  It remains to consider the defender's belief at the information set corresponding to G (i.e., off the equilibrium path). We need to show if the strategy of playing R is optimal given this belief. For this purpose and given that the R is the best response when $\theta \geq \theta^*$, we should calculate the expected payoffs of the defender when it plays R and $\overline{R}$ following strategy G. These payoffs are $q \times -c + (1-q) \times (-\delta - c)$ and $q \times 0 + (1-q) \times 0$. Since there are no values for q that makes the payoff of defender greater for playing R, there is no pooling on (G, G) when the defender plays (R, R).
  Similarly, to verify if there exists a NE where the defender plays $(R, \overline{R})$ we should first show that there are no incentives for the sender to deviate from the pooling (N, N) strategy. This time considering the defender strategy $(R, \overline{R})$, both types of senders do not have any incentive to deviate from N and play G as their payoff would be decreased from $\frac{1}{f}\alpha - \beta$ to $-\beta$ for the bot player and the legitimate user, its payoff remains 0. Now, we should again calculate the payoff of the defender given its belief q. In other words, this time we need to compute the values of q, where $q \times -c + (1-q) \times (-\delta - c) \leq 0$. Hence for all values of $q \leq 0$, the payoff of the defender would be greater when it plays $\overline{R}$. Then, we can conclude that there exists one pooling equilibrium $\{(N, N), (R, \overline{R}), p = \theta, q\}$ for any q in $\mathbb{G}_{cf}$ when $\theta \geq \theta^*$.
- $\theta \leq \theta^* := \frac{\delta+c}{\delta+\alpha(1+\lambda-\frac{1}{f})}$: In this case the best response of the defender following N signal is $\overline{R}$. Similar to the previous case, we should first verify if there is any incentive for the senders to deviate from N, if the defender plays $\overline{R}$ and $\overline{R}$ off the equilibrium path. Since both types will not gain any extra benefits (for bots the payoff decreases from $\alpha - \beta$ to β and for the legitimate users there is no differences), there is no incentives for them to deviate from playing N. Similar calculations can be done for the payoff of the defender, off the equilibrium path to find the possible values for q. Similar to the previous case, there are no values for q, where the expected payoff of playing R would be bigger than $\overline{R}$. Considering all possible deviations for the case that the defender plays $\overline{R}$ and the values of the belief for q there exists another pooling equilibrium, where $\{(N, N), (\overline{R}, \overline{R}), p = \theta, q\}$ for any q in $\mathcal{G}^{cf}$ when $\theta \leq \theta^*$.

□

Theorem 1 represents that if the selected strategy of both types of the senders are N, there is an equilibrium considering the belief of the defender for the possible attacker. In this case, depending on the value of θ, the defender should select one of the strategies R or $\overline{R}$ upon receiving signal N. In other words, if the probability of the sender being a bot (i.e., θ) is greater than θ^*, the defender should run RRM. Otherwise, the best response for the defender is playing $\overline{R}$ strategy. In the case of $\theta \geq \theta^*$ and $\theta \leq \theta^*$, if users send reconnaissance packets, the defender's response will be R and $\overline{R}$, respectively.

Theorem 2. *For any values of θ, there does not exist any pooling equilibrium on G, in $\mathcal{G}^{cf}$ signaling game.*

Proof. In the game $\mathcal{G}^{cf}$ for any values of belief θ, the defender's best respond to pooling strategy of (G, G) is always $\overline{R}$. In other words, the defender does not perform RRM in this case. Consequently, we should see if the senders have any incentive to deviate from G strategy. Let us consider two possible strategies of the defender off the equilibrium path (when it believes in playing with Bot with probability p). Since by deviating from G to N, the sender of type t_1 (Bot) can always increase his/her payoff from $-\beta$ to $\frac{1}{f}\alpha - \beta$ (when the defender plays R off the equilibrium path) or from $-\beta$ to $\alpha - \beta$ (when the defender plays $\overline{R}$ off the equilibrium path), (G, G) cannot be at any pooling equilibrium. □

Theorem 3. *There is no separating equilibrium on (G, N) in the $\mathcal{G}^{cf}$ signaling game.*

Proof. Suppose (G, N) is a pair of senders' strategy, then both of the defender's information sets are on the equilib-

TABLE 2: Equilibria and Their Conditions

Theorem	Conditions	Range of θ	#	$PBNE$	Condition on Beliefs	
					On-equilibrium	Off-equilibrium
Theorem 1	–	$\theta \geq \theta^*$	$\mathcal{PBNE}_1$	$\{(N,N),(R,\overline{R}),p,q\}$	$p=\theta$	$\forall q$
Theorem 1	–	$\theta \leq \theta^*$	$\mathcal{PBNE}_2$	$\{(N,N),(\overline{R},\overline{R},),p,q\}$	$p=\theta$	$\forall q$
Theorem 4	$\lambda \geq \lambda^*$	$\forall \theta^*$	$\mathcal{PBNE}_3$	$\{(N,G),(R,\overline{R}),p,q\}$	$p=1$	$q=0$
Theorem 4	$\lambda \leq \lambda^*$	$\forall \theta^*$	$\mathcal{PBNE}_4$	$\{(N,G),(\overline{R},\overline{R}),p,q\}$	$p=1$	$q=0$

rium path, so both beliefs are determined by Bayes' rule and sender strategy: $q=1$ and $p=0$. Defender's best response following these beliefs is always $\overline{R}$, for both types of the sender. Hence, we should check if the sender's strategy is optimal given the defender strategy. If the sender of type 1 deviates by playing N signal instead of G, the defender responds with $\overline{R}$, giving t_1 (i.e., the Bot) a payoff of $\alpha-\beta$, which exceeds t_1's payoff of $-\beta$ from playing G. Thus, (G,N) cannot establish any separating equilibrium. □

Theorem 4. *There are two classes of separating equilibrium* $\{(N,G),(\overline{R},\overline{R}),p=1,q=0\}$ *if* $\lambda \leq \alpha(1/f-1)+c$ *and* $\{(N,G),(R,\overline{R}),p=1,q=0\}$ $\lambda \geq \alpha(1/f-1)+c$ *in the* $\mathcal{G}^{cf}$ *signaling game.*

Proof. Similar to Theorem 3, we first assume that (N,G) is a pair of senders' strategy. Then both of the defender's information sets are on the equilibrium path: $p=1$ and $q=0$. Considering (N,G) strategy of the senders, the defender's best response following these beliefs can be calculated by comparing the payoffs of the defenders. For sender with type t_1 (i.e., bot), the best response of the defender is calculated by comparing the following two payoffs: $\lambda-\frac{1}{f}\alpha-c$ and $-\alpha$. Two cases can be identified given that $\lambda^*=\alpha(\frac{1}{f}-1)+c$:

- $\lambda \geq \lambda^*$: The best response to play N by type t_1 is R and the best response to type G by type t_2 is $\overline{R}$. We should check, if the sender's strategy is optimal given the defender strategy. Since the bot payoff would be decreased from $\frac{1}{f}\alpha-\beta$ to $-\beta$ and there is no difference for the legitimate payoff, we can conclude that the $\{(N,G),(\overline{R},\overline{R}),p=1,q=0\}$ is a PBNE. We name it as $\mathcal{PBNE}_3$ for later references.
- $\lambda \leq \lambda^*$: In this case, the best response to play N by type t_1 is $\overline{R}$ and the best response to type G by type t_2 is $\overline{R}$. Similar to the previous case there is no incentive for the senders to deviate from (N,G) strategy.

□

The above results present all possible PBNEs of game $\mathcal{G}^{cf}$. Considering these equilibria, we can now provide the best plan of actions for the defender given its belief about the sender's type and its payoff at different strategies.

4.2 Protocol Design

In this section, we design a strategic Crossfire defense mechanism to optimize the strategy of the defender. Table 2 summarizes the results presented in Theorems 1, 2, 3, and 4. All possible separating and pooling PBNEs are displayed. Leveraging these results, we design our proposed strategic Crossfire defense mechanism, namely *Strategic RRM*. The pseudocode of this proposed mechanism is given in Algorithm 1.

Algorithm 1 Strategic Crossfire Defense Mechanism

```
h: Host ID (i.e., its IP address) communicating with the server
t^h_alive: The time that host h is still transmitting packets
f: The frequency at which we can run RRM
P ⇐ 1/f
n := 1
θ_h ⇐ Initialize the belief for h
while ((n − 1) × T) ≤ t^h_alive do
    Estimate/calculate α, β, λ, δ, and c.
    for Each packet received from h between (n − 1)T and nT do
        Update θ_h according to the packet type (signal)
        Compute θ* (Theorem 1)
        Compute λ* (Theorem 4)
    end for
    if λ ≥ λ* then
        Perform RRM
    else if θ ≥ θ* then
        Perform RRM
    end if
    n := n + 1
end while
```

In the Algorithm 1, we first pick a value for the frequency (f) of RRM that represents how often the defender can perform RRM. It is decided through our preliminary experiments. The minimum period of running RRM is shown as P which is assigned to $1/f$. Then the belief for each client is initialized to zero at the beginning of the experiments. The defender always updates its belief each time it receives a packet, and it computes the potential gain/payoff. Later, if the belief is greater than θ^* and the received packet is reconnaissance, the defender chooses to run RRM at the end of P period. In addition to that, if the value of λ is greater than λ^*, the defender should also run the RRM. Hence, the optimal decision for the defender is obtained according to equilibria $\mathcal{PBNE}_1$ and $\mathcal{PBNE}_3$. The same processes after the initialization will be performed every P seconds.

5 Evaluation

In this section, we briefly discuss our experimental setup and evaluation metrics. Then, we present the findings from the experiments.

5.1 Experiment Setup

The network environment is implemented using Mininet [18] and FloodLight [19] is used as an SDN Controller. The proposed signaling game-based defense strategy is developed as an application on top of the FloodLight Controller. In the experimental setup, we consider the network topology with 20 switches, 3 decoy servers, 1 target server, and

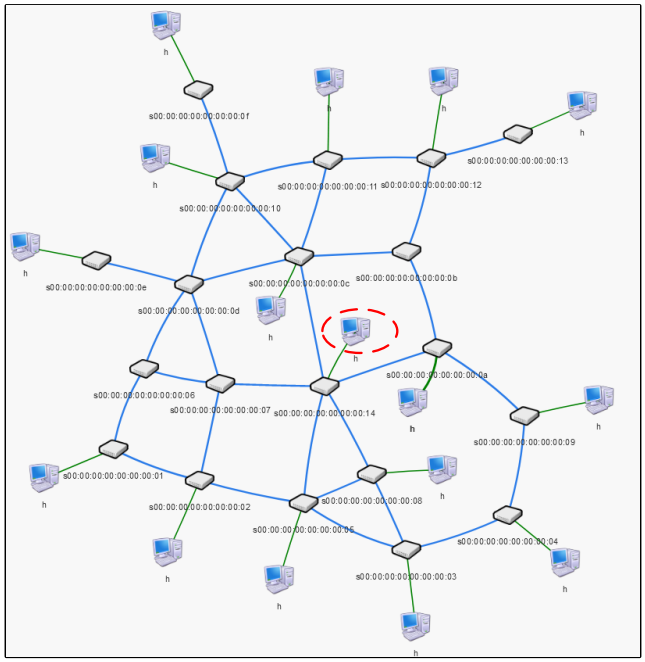


Fig. 4: Experimental Network Topology

several clients, as shown in Fig. 4. The number of clients differs for each experiment based on the attack model and the purpose of the experiment. The target server is encircled by red and decoy servers are placed interior part of the network topology in Fig. 4. The bandwidth of each link in the network is configured as 100 Mbps. A python-based client/server application is implemented for the communication between host-target servers. We implement different attacker models and apply in experiments to evaluate them. The configuration of the network is further explained below.

5.1.1 Defender's Setup

In our experiments, we consider 3 different configurations for the defender. We name them as *No Defense*, *Periodic RRM*, and *Strategic RRM*.

- *No Defense*: To observe the worst-case attack scenario the network environment is implemented without any defense mechanism. In this case, the attacker can easily gain the (static) routing information by doing reconnaissance attacks. Next, s/he attacks the target links which are found by using that static information.
- *Periodic RRM*: The second defense setup is based on performing RRM periodically as suggested in [7], [27]. The routes are changed periodically to the alternative ones that are pre-calculated. We choose different values of frequencies to run RRM in order to observe the impact.
- *Strategic RRM*: It is our proposed signaling game-based defense mechanism. In this case, the defender takes actions based on the attacker's actions as explained in Section 3.2. The mechanism is given in Algorithm 1 and is implemented accordingly.

5.1.2 Attacker's Setup

In a Crossfire attack setup, a decoy server does not receive a large amount of traffic from one or multiple bots simultaneously so that the activity is not considered as suspicious at the network administrator's side. Thus, we limit each bot's communication to only 1 decoy server with a regular amount of traffic. Each bot sends/receives approximately 5 Mbit of data per second in order to behave as legitimate.

We implement two different attacker behaviors, namely *aggressive* and *stealthy*, with two different capabilities, such as *decent* and *strong*, as mentioned in Section 2.3. The aggressive attacker's time for reconnaissance is fixed to 1 minute in experiments. The decent attacker has a limited number of bots, which is as much as the number of legitimate clients in the network. Meanwhile, the strong attacker is equipped with a doubled number of bots. Even though the attack impact is expected to increase, it should also be noted that the more bots the attacker employs, the higher cost is incurred on him/her. We run our experiments on each case to show how effective our solution would be with different attack models.

5.2 Evaluation Metrics

The application of RRM introduces some overhead cost on the defender since it creates extra packets in the network. This overhead can cause Quality of Service (QoS) problems in a form of increased communication latency/packet delay or a higher number of packet losses for the legitimate users. We specifically consider the following metrics to be measured in our experiments:

- *Delay for using longer path:* Using a randomized alternative path can cause longer end-to-end delay since the route may not be the optimal/shortest anymore. This metric shows the increase in delay compared to the optimal path.
- *Delay for flow table updates:* Flow tables are updated whenever RRM is triggered. This process of updating flow table entries of switches takes time. This time is measured to show additional delays that legitimate clients are exposed to. Related to this issue, some of the packets may drop if the queue of a switch becomes full and it cannot handle any more packets whenever the flow table is updated even though no attack is being taken place.
- *The number of packet losses:* This metric represents the number of packets lost for legitimate users due to attacks or the execution of RRM.

5.3 Experimental Results

We run experiments on different network configurations to observe each evaluation metric separately. We first show the overhead of performing RRM on the network (i.e., legitimate users) varying the RRM frequency, i.e., the interval period between two subsequent route mutations. We find the optimal period among them, and use this *Periodic RRM* to compare with the proposed *Strategic RRM* by running experiments with different adversary models.

5.3.1 Observing RRM Costs without any Attacker

We run our experiments without considering any malicious activities in the network in order to measure the defender's

TABLE 3: Delay Caused by RRM (in microsecond)

Client ID	*Delay Without RRM*	*Delay with RRM*					
		Optimum size Path			*Any size Path*		
		60-seconds RRM	*30-seconds RRM*	*10-seconds RRM*	*60-seconds RRM*	*30-seconds RRM*	*10-seconds RRM*
1	2853	2892	3040	3398	3781	3717	4053
2	2854	2898	3050	3406	3761	3616	4145
3	1497	1533	1629	1882	1973	2075	2351
4	1499	2161	2322	2698	2986	3205	3578

TABLE 4: Packet Drop Caused by Flow Table Update

10-seconds RRM	30-seconds RRM	60-seconds RRM
10.1%	8.6 %	1.6%

cost (i.e., the cost imposed on its clients). In the experiments, we measure the additional delay that is caused due to updating flow tables as well as using longer paths when RRM is applied. We first change the RRM frequency to observe the flow table update cost. Here, the routing paths are kept the same. Then, to assess the delay for using longer paths, we run experiments selecting optimal (shortest length) paths as well as non-optimal (alternative) paths.

Table 3 presents the average packet delay for 4 randomly selected legitimate clients. We can easily see that a higher frequency of RRM (i.e., when RRM interval period is 10 seconds) brings additional delay to the clients if alternative path sizes are the same as the optimum path. The slight difference between each column is caused by flow table updating. As an example, we can take a look at the delays for Client 1: during the 60-seconds period RRM, the average delay is 2892 milliseconds, while the delay becomes 3040 milliseconds and 3398 milliseconds when the periods are 30-seconds and 10-seconds, respectively. Even though the difference between them looks negligible (in milliseconds), it should be noted that the average delay is computed for approximately 100 thousands of packets in a 10-minute long experiment. Thus, the sum of additional delays that are caused by flow table updates, is a significant cost considering the delay increase for 100 thousands of packets where only a few times flow table updates are performed. We can also observe in Table 3 that if we use alternative (i.e., non-optimal) paths for clients, the average delay increases significantly. This is because the increased number of hops adds further delay to the packet communication.

Next, we assess if the route updates can cause packet drops. The experimental setup remains the same as the previous ones except we consider a higher number of clients to send more simultaneous packets in order to occupy the queues/buffers at the switches and simulate an environment that would more likely occur during Crossfire attacks. We use small sized packets (100 bytes each) so that the link bandwidth cannot be a cause of packet dropping. The results from the experiments are presented in Table 4. We observe that a higher frequency of RRM causes a larger number of packet drops. This is reasonable considering switches are busier updating their flow tables in the cases where more often route changes are occurred. In other words, some of the packets cannot be handled on time and are dropped because of overflowing of the queue.

The 60-seconds RRM case causes fewer delays and less packet loss compared to 30-seconds and 10-seconds RRM cases if there is no attack, as shown in Tables 3 and 4. In other words, the 10-seconds RRM case causes lots of overhead compared to 60-seconds RRM. Thus, we opt to run 60-seconds RRM in the following experiments to compare with *Strategic RRM*.

5.3.2 Comparing Strategic RRM with Periodic RRM

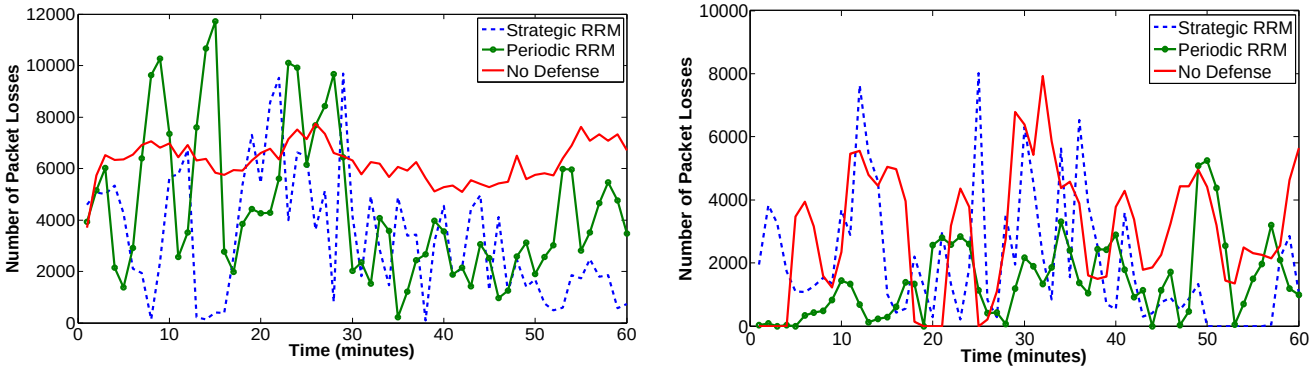
The values for different game parameters of *Strategic RRM* are chosen as shown in Table 6. These parameters are observed and tuned based on our preliminary experiments on the RRM cost and the impact of attacks. More specifically, α is the attacker's gain, and it is considered as the percentage of packet losses due to the attack (when there is *No Defense* mechanism), which is 3% where the attacker is stealthy and decent, as shown in Table 5. This value increases if the attacker's strategy is more intense or network size is smaller. Parameter λ is the gain of the defender that s/he earns by defending against the reconnaissance attack. It is calculated by deducting 0.7% packet loss when the 60-seconds *Periodic RRM* is applied from 3% packet loss when there is *No Defense* mechanism. Thus, λ is set at 2.3. λ increases if RRM is applied more often than 60-seconds. The parameter δ is the cost of the defender with respect to the legitimate users. We derive this parameter as the percentage of increase in the average packet delay from the case of *No Defense* (a delay of 2175 microseconds) to the case of 60-seconds RRM (a delay of 2371 microseconds). Hence, we calculate δ as 9. The value of δ would increase if RRM is applied more often. In addition, we assign 1.6 to c since it is the packet loss (as shown in Table 4) in the case of 60-seconds RRM. The cost variable, c, increases if RRM is more frequent.Finally, we consider the minimum possible interval of running RRM as 1 second. Hence, the frequency (f) is set to 1.

In *Strategic RRM*, the defender builds its belief about each of its clients of being a bot and perform RRM accordingly as it is mentioned in Section 3.3. In other words, the attacker is able to run a stealthy attack initially until the belief increases to some extent. Therefore, it is expected to see more packet losses for *Strategic RRM* at the beginning of the experiments. Meanwhile, it is also expected that *Periodic RRM* causes a similar number of packet losses since it has the same strategy throughout the experiments.

In the next experiments, we implement a different attack models for each defender type and run them separately. In order to observe the performance in different attack models, we plot the graphs individually in Fig. 5 and Fig. 6. Packet loss percentage and average delay for all network nodes are given in Table 5. The attacker model demonstrated in Fig. 5a considers a strong one with an aggressive behavior. As the Fig. 5a shows, in the case of *No Defense*, the number of packet losses are high and it stays

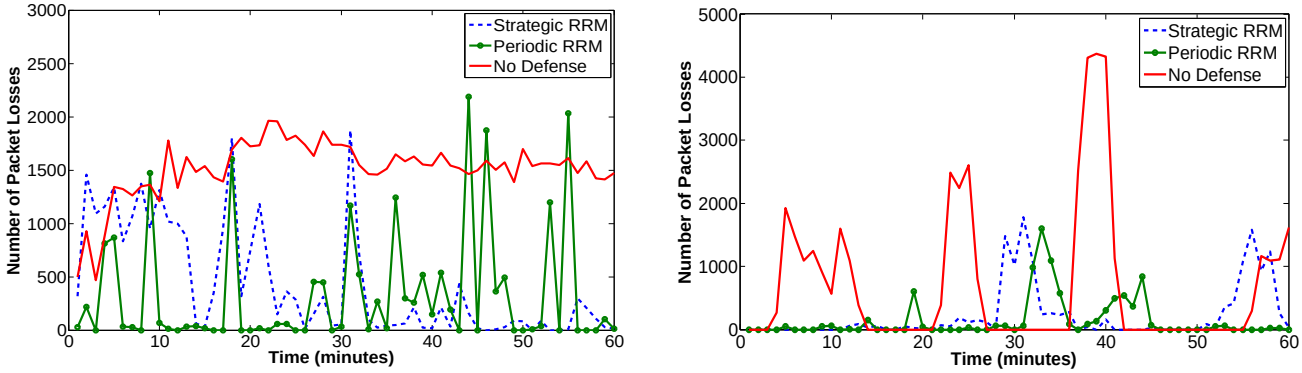
TABLE 5: Cost Metric for Different Attacker Model

Attacker Model	Packet Loss			Average Delay (in milliseconds)		
	No Defense	Periodic RRM	Strategic RRM	No Defense	Periodic RRM	Strategic RRM
Strong-Aggressive	31%	22%	16%	1172	989	438
Strong-Stealthy	15%	7%	10%	579	380	278
Decent-Aggressive	7%	2%	2%	464	251	200
Decent-Stealthy	3%	0.7%	1%	263	168	141



(a) Aggressive Behavior Model (b) Stealthy Behavior Model

Fig. 5: Packet Loss with Strong Attacker



(a) Aggressive Behavior Model (b) Stealthy Behavior Model

Fig. 6: Packet Loss with Decent Attacker

TABLE 6: Parameters Used in Simulation

Parameter Name	α	λ	δ	c	f
Value	3	2.3	9	1.6	1

almost similar throughout the experiments. However, in the cases of *Periodic RRM* and *Strategic RRM* mechanisms, the scenario changes dramatically. *Strategic RRM* has a much less number of packet losses after the first few minutes of the experiment. This initial period takes the belief to a state at which RRM starts to perform properly against the possible bots. An important observation is that *Periodic RRM* usually has a higher number of packet losses continuously since it changes the routes at every time interval of the period which may not overlap with the attacker's reconnaissance phase. However, this is not the case for *Strategic RRM*. It defends against the attack only when there are highly suspicious activities, which is reflected by the increased belief and the equilibrium conditions. Another interesting observation in Fig. 5a is that the peak number of packet losses appears at the beginning of the RRM cases, not in the *No Defense* case. This can be explained as the nature of RRM. Since it is based on random mutation, it may cause more routes to the same link which can end up congesting that specific link intensively. However, in *No Defense* case, the attacker can only target the same link set unless new bots are added and new routing paths are identified. In this attack model, the overall network's packet loss percentages are 31% for *No Defense*, 22% for *Periodic RRM*, and 16% for *Strategic RRM* as presented in Table 5. Average transmission delays are parallel to the packet loss results and decrease from 1172 milliseconds at *No Defense* case to 989 milliseconds *Periodic RRM* case, and to 438 milliseconds in *Strategic RRM* case.

In addition to the attacker with an aggressive behavior, we also consider a stealthy one and represent the results in Fig. 5b. In this graph, we can see unbalanced results which are basically due to the attacker's dynamic behavior. The stealthy attacker model can stay idle some of the time

and can do longer reconnaissance attacks. Hence, it leads to such uneven consequences. Unlike the aggressive attack model, *Strategic RRM* has higher packet losses (i.e., 10%) compared to *Periodic RRM* case's packet loss (i.e., 7%), even after the belief gets time to be matured as shown in Table 5. The main reason behind this behavior is the characteristic of the stealthy attacker model where the attacker reduces and increases the attack intensity randomly making the defender's belief change more often. Even though *Strategic RRM* has higher packet loss, the average transmission delay is 278 milliseconds for the *Strategic RRM* case while it is 380 milliseconds in *Periodic RRM*. Moreover, it should be noted that *Periodic RRM* brings additional overhead to the network even if there is not an attack targeting the network. Hence, we claim that it is reasonable to use *Strategic RRM* considering the overall advantages.

Furthermore, we run the same experiments with a decent attacker. As we defined earlier, the number of bots he has is half of the bots that the aggressive attacker has. The results are shown in Fig. 6, which are correlated with the ones in Fig. 5. One significant observation is that the numbers of packet losses are decreased by more than half. In other words, increasing the number of bots, as in the aggressive case, raises the damage significantly, more than a linear increase. It is easily noticed that the number of packet losses goes down to zero in the case of the decent attacker as shown in Fig. 6a and Fig. 6b. This is because there are fewer bots to be protected against. The other result to pay attention is that there is a more high number of packet losses consecutively in *Strategic RRM* compared to a strong attacker since less number of bots create less suspicion compared to more bots as it is shown in Equation (1). This interesting outcome can be seen in the comparison of Fig. 5b and Fig. 6b. In addition to that, *Strategic RRM* has almost the same overall contribution (e.g., 2% for aggressive, and 1% for stealthy) to the network defense compared to *Periodic RRM* (e.g., 2% for aggressive, and 0.7% for stealthy) as reported in Table 5. Yet, there is still a notable transmission delay decrease from 251 milliseconds to 200 milliseconds with Aggressive Attacker, and from 168 milliseconds to 141 milliseconds with Stealthy Attacker whenever *Strategic RRM* is used.

In addition to the given metrics, we also considered boundaries of the network defense in terms of the maximum number of attackers that can be tolarated. To calculate this metric, we used the network characteristics that we have implemented in our experiments. Specifically, our network links are 100Mbps, and there are five different network links to access the data server. Therefore, the total traffic to congest all routes is equal to 500Mbps. Each attacker creates and transmits 5Mbps of data. Thus, our MTD-based defense is able to keep the network functional until 100 bots attack simultaneously. We should note that 100 bot requirement is a high cost for attacking a very small-scale network and this number increases linearly with the network size.

6 Related Work

In this section, we talk about other's works in a similar area. We first explain research papers proposing defense mechanisms against SLFAs, and later we discuss signaling games paper for cybersecurity.

6.1 Defense Mechanisms for SLFA

There are various defense mechanisms proposed for the SLFA in literature according to the recent survey on the LFA in SDN ecosystems [28]. Only a few of them consider defending against reconnaissance phase of the attack [13], [29], [30] while most of them strive to protect the network during the data phase of the attack [6], [10], [11], [12], [31], [32]. Authors in [29] claim traceroute packets are increased before an SLFA occurs, and they design a detection mechanism for the attacker based on that assumption. Similarly, in [33] authors utilize traceroute packet correlation by applying machine learning techniques to detect SLFA in the Internet of Things (IoT) environment. Even though we also have a similar belief that traceroute packets will increase before the attack, we do not rely on this feature individually, and our attack mitigation technique is different. Meier et al. obfuscate the attacker's reconnaissance by replying with some virtual hops that are consistent with the network topology [13]. The other work proposes to confuse the attacker by utilizing honeypots in SDN [30]. While these solutions aim to prevent the attacker from gaining critical information, they give wrong information to legitimate users. We argue that these approaches are not the best practice since a legitimate user needs to get the right information in order to utilize the network facilities in the most efficient way.

Meanwhile, authors in [11] propose Traffic Engineering (TE) as a solution to mitigate an SLFA . In their solution, the defender forces the attacker to use the improbable path (i.e., very unlikely to be used) so that the attacker ends up being identified. Similarly, observing traffic patterns while attack happens is applied to detect and defer an SLFA in [10]. The main problem with these solutions is that they are reactive and some critical harm could have been done before the attack is mitigated. Traffic engineering based solution is also used by authors in [31]. The authors suggest upgrading switches to SDN-based switches in order to detect and mitigate an SLFA. However, it is not specified how to upgrade switches in run-time and how SDN switches are capable of detecting such attacks. In [32], even though collaboration between ASes is shown helpful whenever an SLFA hits, it is not clear how to manage different ASes to work together. [12] observes link-probers by checking the packets at the ingress port. If a sender is found as link-prober, then Linkbait applies MTD to confuse its route. They suggest matrix-based feature extraction in order to detect which link-prober is a bot. In [6], authors propose utilizing SDN capabilities to detect the attacker's activities and block the malicious activity. While our solution is similar to [12] and [6], we do not rely on identifying bots which could not be possible with an intelligent adversary model. Differently, we have designed a signaling game which lets the defender to decide which action to take considering his/her payoff. Authors in [34] utilizes a very similar approach by modeling the attacker and the defender into a Stackelberg game. However, their main idea behind the defense strategy is the detection of congestion in the links, which is reactive, and the defense and mitigation might be late to hinder some packet losses. Our solution is different due to our pre-attack based approach.

6.2 Signaling Games for Cybersecurity

In many cybersecurity scenarios, the defender cannot clearly detect whether the received messages are coming from a benign user or they are a part of the attack scenario and are initiated by attackers. Signaling game is a two-player incomplete information game that has been used to model different cybersecurity problems, such as intrusion detection [35] or deception [36]. Furthermore, the authors model cybersecurity in terms of signaling games in [37] and W. Casey et al. model deception with signaling game in [38]. They present how signaling games provide a formal mathematical method to analyze the way of identity and deception coupling in cyber-social systems. The game-theoretic framework can be extended to reason about dynamical system properties and behavior traces. In [39], the authors formulate a deception with signaling game in networks in which the defender deploys a fake avatar for identification of the compromised internal user. In [40], the authors propose a selective and dynamic mechanism for counter-fingerprinting. They model and analyze the interaction between a fingerprinter and a target as a signaling game. Following this work in [41], the authors suggest changing attack surface (e.g., port numbers) depending on a belief that is observed in the signaling game. In [42], the authors investigate the interactions between a service provider and a client by signaling game, where the client does not have complete information about the security conditions of the service provider. In [43], the authors propose a moving target-based deceptive defense mechanism using a signaling game for the frequency of migrations of the virtual machines in clouds. While we also apply signaling game for network defense, we propose using RRM which has not been studied under a signaling game. Furthermore, our framework is applied not only to reconnaissance attacks but also real data phase of the attacks. The reason behind choosing and applying the signaling game approach specifically for SLFA is that we believe the dynamic and unnoticeable behavior of SLFA can be modeled into a signaling game accurately.

7 Conclusion

While the online services are rapidly growing in number, DDoS attacks have become very common. International criminals also utilize sophisticated cyber attacks to steal information and take down critical infrastructure. Therefore network administrators need to deploy methods to get engaged with attackers to have a better understanding of attacker intentions/behaviors and improve defense success for future attacks. Our work is a demonstration of this perspective. In this paper, we present a signaling game-based dynamic MTD to defend against Crossfire attacks. We first model the attacker and the defender as a signaling game. Considering their payoffs, we compute the equilibria of the game, which represent the best strategies for each player considering the opponent is rational. According to the game results, we develop an algorithm, namely *Strategic RRM*. We implement and compare it with *Periodic RRM*. Our experimental results show that *Strategic RRM* can lessen the impact of Crossfire attacks similar to *Periodic RRM*, while it brings significantly less overhead. In the future, we would like to extend our strategic MTD-based framework for other DDoS-based emerging attacks.

References

[1] Netscout, "14th annual worldwide infrastructure security report (wisr)," 2018. [Online]. Available: https://www.netscout.com/report/

[2] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks," *IEEE communications surveys & tutorials*, vol. 15, no. 4, 2013.

[3] A. Studer and A. Perrig, "The coremelt attack," in *European Symposium on Research in Computer Security*. Springer, 2009, pp. 37–52.

[4] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *Security and Privacy (SP), IEEE Symposium on*, 2013, pp. 127–141.

[5] S. Venkatesan, M. Albanese, G. Cybenko, and S. Jajodia, "A moving target defense approach to disrupting stealthy botnets," in *Proceedings of the ACM Workshop on Moving Target Defense*, 2016.

[6] J. Wang, R. Wen, J. Li, F. Yan, B. Zhao, and F. Yu, "Detecting and mitigating target link-flooding attacks using sdn," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 6, 2018.

[7] A. Aydeger, N. Saputro, K. Akkaya, and M. Rahman, "Mitigating crossfire attacks using sdn-based moving target defense," in *41st Conference on Local Computer Networks (LCN)*. IEEE, 2016.

[8] A. Aydeger, N. Saputro, and K. Akkaya, "A moving target defense and network forensics framework for isp networks using sdn and nfv," *Future Generation Computer Systems*, vol. 94, pp. 496–509, 2019.

[9] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Formal approach for route agility against persistent attackers," in *European Symposium on Research in Computer Security*. Springer, 2013, pp. 237–254.

[10] L. Xue, X. Ma, X. Luo, E. W. Chan, T. T. Miu, and G. Gu, "Linkscope: toward detecting target link flooding attacks," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2423–2438, 2018.

[11] D. Gkounis, V. Kotronis, C. Liaskos, and X. Dimitropoulos, "On the interplay of link-flooding attacks and traffic engineering," *ACM SIGCOMM Computer Communication Review*, vol. 46, no. 2, 2016.

[12] Q. Wang, F. Xiao, M. Zhou, Z. Wang, Q. Li, and Z. Li, "Linkbait: Active link obfuscation to thwart link-flooding attacks," *arXiv preprint arXiv:1703.09521*, 2017.

[13] R. Meier, P. Tsankov, V. Lenders, L. Vanbever, and M. Vechev, "Nethide: secure and practical network topology obfuscation," in *27th {USENIX} Security Symposium*, 2018, pp. 693–709.

[14] A. NOC, "How to: Mtr – understanding and troubleshooting network connectivity," 2015. [Online]. Available: https://www.atlantic.net/vps-hosting/how-to-mtr-understanding-troubleshooting-network-connectivity/

[15] G. Antichi, M. Shahbaz, Y. Geng, N. Zilberman, A. Covington, M. Bruyere, N. McKeown, N. Feamster, B. Felderman, M. Blott *et al.*, "Osnt: Open source network tester," *IEEE Network*, vol. 28, no. 5, pp. 6–12, 2014.

[16] W. Tolliver, "Disabling icmp and snmp won't increase security, but will impact network monitoring," 2020. [Online]. Available: https://blog.paessler.com/disabling-icmp-and-snmp-wont-increase-security-but-will-impact-network-monitoring

[17] T. H. Noe, "Capital structure and signaling game equilibria," *The Review of Financial Studies*, vol. 1, no. 4, pp. 331–355, 1988.

[18] M. Team, "Mininet," 2016. [Online]. Available: http://mininet.org/

[19] F. Project, "Floodlight controller," 2014. [Online]. Available: http://www.projectfloodlight.org/floodlight/

[20] P. K. Manadhata and J. M. Wing, "An attack surface metric," *IEEE Transactions on Software Engineering*, no. 3, pp. 371–386, 2010.

[21] P. K. Manadhata, "Game theoretic approaches to attack surface shifting," in *Moving Target Defense II*. Springer, 2013, pp. 1–13.

[22] N. McKeown, "Software-defined networking," *INFOCOM keynote talk*, vol. 17, no. 2, pp. 30–32, 2009.

[23] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[24] R. S. Ramanujan, M. N. Kaddoura, X. Wu, and K. S. Millikin, "Protecting networks from access link flooding attacks," Apr. 8 2008, uS Patent 7,356,596.

[25] Y. Shoham and K. Leyton-Brown, *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[26] R. Gibbons *et al.*, "A primer in game theory," 1992.

[27] Q. Duan, E. Al-Shaer, and H. Jafarian, "Efficient random route mutation considering flow and network constraints," in *2013 IEEE Conference on Communications and Network Security (CNS)*, pp. 260–268.

[28] R. ur Rasool, H. Wang, U. Ashraf, K. Ahmed, Z. Anwar, and W. Rafique, "A survey of link flooding attacks in software defined network ecosystems," *Journal of Network and Computer Applications*, p. 102803, 2020.

[29] T. Hirayama, K. Toyoda, and I. Sasase, "Fast target link flooding attack detection scheme by analyzing traceroute packets flow." in *WIFS*, 2015, pp. 1–6.

[30] J. Kim and S. Shin, "Software-defined honeynet: Towards mitigating link flooding attacks," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 99–100.

[31] L. Wang, Q. Li, Y. Jiang, and J. Wu, "Towards mitigating link flooding attack via incremental sdn deployment," in *2016 IEEE Symposium on Computers and Communication (ISCC)*, pp. 397–402.

[32] S. B. Lee, M. S. Kang, and V. D. Gligor, "Codef: collaborative defense against large-scale link-flooding attacks," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, 2013, pp. 417–428.

[33] X. Lixia, D. Ying, Y. Hongyu, and H. Ze, "Mitigating lfa through segment rerouting in iot environment with traceroute flow abnormality detection," *Journal of Network and Computer Applications*, vol. 164, p. 102690, 2020.

[34] X. Ma, B. An, M. Zhao, X. Luo, L. Xue, Z. Li, T. Miu, and X. Guan, "Randomized security patrolling for link flooding attack detection," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 4, 2019.

[35] S. Shen, Y. Li, H. Xu, and Q. Cao, "Signaling game based strategy of intrusion detection in wireless sensor networks," *Computers & Mathematics with Applications*, vol. 62, no. 6, pp. 2404–2416, 2011.

[36] J. Zhuang, V. M. Bier, and O. Alagoz, "Modeling secrecy and deception in a multiple-period attacker–defender signaling game," *European Journal of Operational Research*, vol. 203, no. 2, 2010.

[37] W. Casey, J. A. Morales, T. Nguyen, J. Spring, R. Weaver, E. Wright, L. Metcalf, and B. Mishra, "Cyber security via signaling games: Toward a science of cyber security," in *International Conference on Distributed Computing and Internet Technology*. Springer, 2014.

[38] W. Casey, A. Kellner, P. Memarmoshrefi, J. A. Morales, and B. Mishra, "Deception, identity, and security: the game theory of sybil attacks," *Communications of the ACM*, vol. 62, no. 1, 2018.

[39] A. Mohammadi, M. H. Manshaei, M. M. Moghaddam, and Q. Zhu, "A game-theoretic analysis of deception over social networks using fake avatars," in *International Conference on Decision and Game Theory for Security*. Springer, 2016, pp. 382–394.

[40] M. A. Rahman, M. H. Manshaei, and E. Al-Shaer, "A game-theoretic approach for deceiving remote operating system fingerprinting," in *Conference on Communications and Network Security (CNS)*. IEEE, 2013, pp. 73–81.

[41] Z. Zhao, F. Liu, and D. Gong, "An sdn-based fingerprint hopping method to prevent fingerprinting attacks," *Security and Communication Networks*, 2017.

[42] M. M. Moghaddam, M. H. Manshaei, and Q. Zhu, "To trust or not: a security signaling game between service provider and client," in *International Conference on Decision and Game Theory for Security*. Springer, 2015, pp. 322–333.

[43] M. T. Adili, A. Mohammadi, M. H. Manshaei, and M. A. Rahman, "A cost-effective security management for clouds: A game-theoretic deception mechanism," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*.

Abdullah Aydeger is an Assistant Professor in the Department of Computer Science at Southern Illinois University, Carbondale. He received his M.S. and a Ph.D. degree from the Department of Computer Engineering at Florida International University in 2016 and 2020. His research interests include Software Defined Networking, Network Function Virtualization, Moving Target Defense, and their utilization for network security and resiliency. His expertise is on link flooding attacks. He applies the ideas not only to traditional ISP networks but also to emerging network domains within cyber-physical systems and IoT. He has published papers in reputable journals and conferences. He has also contributed two book chapters. Mr. Aydeger is a member of IEEE and served as a reviewer for many conferences and journals.

Mohammad Hossein Manshaei received the B.Sc. degree in electrical engineering and the M.Sc. degree in communication engineering from the Isfahan University of Technology, Iran, in 1997 and 2000, respectively. He received another M.Sc. degree in computer science and the Ph.D. degree in computer science and distributed systems from the University of Nice, Sophia-Antipolis, France, in 2002 and 2005, respectively. He did his thesis work at INRIA, Sophia-Antipolis. From 2006 to 2011, he was a Senior Researcher and Lecturer with the Swiss Federal Institute of Technology, Lausanne (EPFL). He held visiting positions at UNCC, NYU, VTech, and UTSA. He is a visiting faculty at the Florida International University and an Associate Professor with the Isfahan University of Technology. His research interests include wireless networking, network security and privacy, computational biology, and game theory.

Mohammad Ashiqur Rahman is an Assistant Professor in the Department of Electrical and Computer Engineering at Florida International University (FIU), USA. He is leading the Analytics for Cyber Defense (ACyD) Lab at FIU. Before joining FIU, he was an Assistant Professor at Tennessee Tech University. He obtained the PhD degree in computing and information systems from the University of North Carolina at Charlotte in 2015. Dr. Rahman's research focus primarily includes artificial intelligence-based novel analytics design and development for network and information security, control-aware resiliency, and security hardening. He has already published over 75 peer-reviewed journals and conference papers. He has served on the technical programs and organization committees for various IEEE and ACM conferences. Dr. Rahman is a senior member of IEEE and a member of ACM.

Kemal Akkaya is a full professor in the Department of Electrical and Computer Engineering at Florida International University. He received his PhD in Computer Science from University of Maryland Baltimore County in 2005 and joined the department of Computer Science at Southern Illinois University (SIU) as an assistant professor. Dr. Akkaya was an associate professor at SIU from 2011 to 2014. He was also a visiting professor at The George Washington University in Fall 2013. Dr. Akkaya leads the Advanced Wireless and Security Lab (ADWISE) in the ECE Dept. His current research interests include security and privacy, internet-of-things, and cyber-physical systems. Dr. Akkaya is a senior member of IEEE. He is the area editor of Elsevier Ad Hoc Network Journal and serves on the editorial board of IEEE Communication Surveys and Tutorials. He has received "Top Cited" article award from Elsevier in 2010.