Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 7264269, 11 pages
https://doi.org/10.1155/2018/7264269

Research Article

ABS-DDoS: An Agent-Based Simulator about Strategies of Both DDoS Attacks and Their Defenses, to Achieve Efficient Data Forwarding in Sensor Networks and IoT Devices

Franks González-Landero,[1] Iván García-Magariño,[2,3] Raquel Lacuesta,[2,3] and Jaime Lloret[4]

[1]*Edison Desarrollos, Teruel 44002, Spain*
[2]*Department of Computer Science and Engineering of Systems, University of Zaragoza, Teruel 44003, Spain*
[3]*Instituto de Investigación Sanitaria Aragón, Zaragoza, Spain*
[4]*Instituto de Investigación para la Gestión Integrada de Zonas Costeras, Universitat Politecnica de Valencia, Valencia, Spain*

Correspondence should be addressed to Iván García-Magariño; ivangmg@unizar.es

Received 23 March 2018; Accepted 28 May 2018; Published 24 June 2018

Academic Editor: Wei Wang

Copyright © 2018 Franks González-Landero et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Sensor networks and Internet of Things (IoT) are useful for many purposes such as military defense, sensing in smart homes, precision agriculture, underwater monitoring in aquaculture, and ambient-assisted living for healthcare. Efficient and secure data forwarding is essential to maintain seamless communications and to provide fast services. However, IoT devices and sensors usually have low processing capabilities and vulnerabilities. For example, attacks such as the Distributed Denial of Service (DDoS) can easily hinder sensor networks and IoT devices. In this context, the current approach presents an agent-based simulation solution for exploring strategies for defending from different DDoS attacks. The current work focuses on obtaining low-consuming defense strategies in terms of processing capabilities, so that these can be applied in sensor networks and IoT devices. The experimental results show that the simulator was useful for (a) defining defense and attack strategies, (b) assessing the effectiveness of defense strategies against attack ones, and (c) defining efficient defense strategies with low response times.

1. Introduction

Sensor networks (SNs) are becoming useful in a large variety of sensing applications. One of these application is to monitor crop fields to irrigate and fumigate some specific areas when necessary [1]. In addition, underwater SNs are useful for assessing amounts of fish in fish farms [2]. This can be useful for delivering the right amount of food for properly feeding fishes in aquaculture without generating unnecessary food wastage. SNs can also be useful for military tactics [3]. In addition, Internet of Things (IoT) is useful for improving lifestyles, automating services, and making more information available in real-time. For example, IoT can be useful for providing the appropriate healthcare of patients when sleeping by means of smart beds [4]. In addition, IoT is also useful for improving the performance of waste collection considering risky material, in smart cities [5].

In general, efficient data forwarding is one of the key features of SNs and IoT devices. However, this efficiency can be hindered by external attacks. Cyber-crimes can provoke damage to normal citizen, companies, and even states. Cyber security measures are necessary to prevent these attacks. A common attack is the Distributed Denial of Service (DDoS) [6]. This attack consists in performing a high number of petitions to a service provider including a sensor or an IoT device, from machine multiples in order to make the target overloaded. There are several ways of performing this attack. The way most common way is to use botnets [7]. Botnets are infected machines whom owners do not know that they are part of an attack. Like in other technological aspects, cyber-attacks are continuously evolving and it is hard to predict what will be the future trends. IoT may cause an increase of infected devices numbers [8]. It is only enough to infiltrate a

harmful agent in a device and without the user knowing. The device may send information about its owner to other sites or the device may self-involve in a DDoS attack. The damage that these attacks can produce are well-known, such as millionaire losses, making an online service inaccessible, and damaging corporate image of a company. This damage motivates the improvement of cyber security techniques. In the context of SNs and IoT normally the processing capabilities are low, and consequently these techniques should be efficient.

The literature also includes both (a) works that focus on specific repercussions of DDoS, and (b) more general approaches that cover DDoS among other attacks. For example, the DDoS attack can be intended to meltdown a data center. In particular, [9] simulated this kind of attack, in which DDoS could be combined with problems/attacks in ventilation or air condition. Hence, this simulator focused on the specific repercussion on heating from DDoS attacks. In a more general context, [10] presented a multilayer approach that defended from multiple kinds of attacks, including DDoS. It exploited the complementary features among different filters obtaining a hybrid approach with low redundancies. However, those works did not provide a mechanism for defining and simulating strategies of DDoS attacks based on different mechanisms of coordination, as the current work does. Those works neither provided the possibility of determining and assessing defense strategies from DDoS attacks, while the current approach supported this possibility.

DDoS attacks can be prevented by defining lightweight algorithms that determine whether a request is real or faked. For example, a lightweight algorithm was defined for protecting controllers and switches in software-defined networks (SDNs) from DDoS attacks [11]. This algorithm was based on the analysis of the packets sent to a SDN and performed significantly better for SDN ecosystems of mobile users.

In this context, the current approach addresses the definition and assessment of both cyber-attacks and cyber-defenses, in order to estimate the cyber-defense's effectiveness when a SN node or a IoT device is attacked in different manners. More concretely, the current approach mainly focuses on the different strategies for performing and defending from DDoS attacks. In this work, we present the novel agent-based simulator (ABS) called ABS-DDoS. This simulator allows engineers to define strategies for performing attacks in different coordinated ways. It also allows engineers to define strategies for estimating which are the attackers in order to deny them the services and consequently being able to provide services the real requests. The current simulator assesses these strategies by simulating these together and providing such as the percentage of real requests that are successfully attended.

2. Materials and Methods

The main material of the current work is the novel simulator about DDoS attacks in sensors and IoT devices called ABS-DDoS, which is presented in Section 2.1. In addition, Section 2.2 describes the strategies that we have defined with ABS-DDoS for the current experiments. Furthermore, Section 2.3 introduces the procedure that we have followed to assess the utility of this novel simulator in improving the security regarding DDoS attacks.

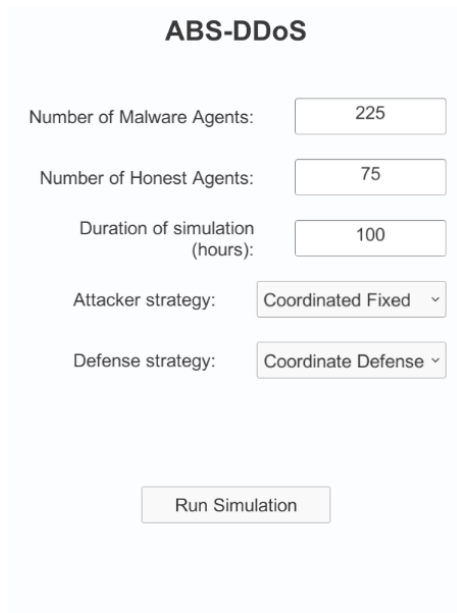


Figure 1: Main screen of the application.

2.1. ABS-DDoS: An ABS of Strategies for Both Performing and Defending from DDoS Attacks. ABS-DDoS is a simulator about DDoS attacks. It is implemented as an ABS, in which sensor and IoT devices are modeled as agents providing services. Other agents coordinately perform DDoS attacks. ABS-DDoS allows users to define and simulate several strategies about DDoS attacks. It also allows users to define defense strategies and simulate their results when defending from certain strategies of DDoS attacks.

Figure 1 presents the main screen of the user interface (UI) of the simulator. This input screen allows users to set the input parameters for simulations, which are (1) number of malware agents, (2) number of honest agents, (3) duration of simulation, (4) attacker strategy, and (5) defense strategy. The first parameter is the number of malware agents and their role is to simulate bots' attacks against a server. The second parameter is the number of honest agents, which simulate requests made by normal users. The third parameter is duration of simulation in h and represents the number of iterations in the simulation. Finally, the last parameters represent attack/defense strategies that users can use in simulations. When a user finalizes setting up all input parameters, they can press "Run Simulation" button to start a simulation.

Figure 2 depicts the whole app functionality. The tool has four main agent types. The simulation entails a periodic execution with several cycles. The number of cycles is established by the duration parameter in the simulation. For instance, if a user sets a duration value of 25, it means that each agent has the possibility of taking autonomous decisions 25 times. Agent subtypes share similar characteristics but

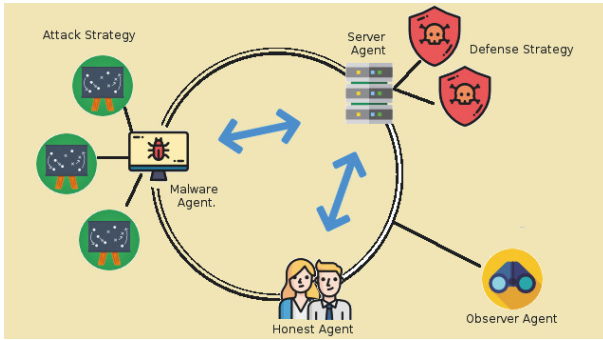


Figure 2: Overview of the strategy system.

they can have their own behaviors. The simulator creates all agents then adds them to simulation and finally executes each agent. The server agent simulates a machine whose function can be to give services, to response to queries, or to forward data. Moreover, server agent represents the target machine that malware agents will attack. The agent server has a certain strategy defense established by the defense strategy parameter. The defense strategy indicates how a server agent defends itself from attacks of malware agents. The malware agent simulates each machine that may perform DDoS attacks. Each malware agent subtype has a certain attack strategy and it describes how to perform attacks against a server agent. Attacks may be in waves, during an elapsed time or coordinated.

The third main agent type is honest agents. These agents represent normal users, SN nodes, or IoT devices that perform legitimate requests to the server agent. The decisions of honest agents about whether requesting services is simulated using a low and configurable value. In particular, we simulated these nondeterministic decisions using the principles of TABSAOND (a technique for developing agent-based simulation apps and online tools with nondeterministic decisions) [12]. A random number is generated in the [0, 1) interval, and it is compared with the probability. If the number is lower than the probability, then the honest agent requests a service to the server agent.

It is worth mentioning that each server agent has a limited number of requests that can be attended per iteration. If the limit is reached, this agent will mandatorily deny all the remaining requests in the corresponding iteration.

The last agent type is observer agent. The main function of this agent is to collect data about simulations. Specifically, it gives us information such as percentage of success of honest agent, percentage of success attackers, and percentage of success of customers in each of iteration. All this information is saved into a file, so we were able to further analyze if after the simulation.

ABS-DDoS was developed with Unity 5.6.1f1. Unity game-based engine is popular and well-known among developers' community. Unity is popular because it is multiplatform, allowing the deployment of applications in several operative systems, typing code only once. We used Unity because it offers a suitable environment in order to work with the Process for developing Efficient Agent-Based Simulators (PEABS) [13]. The underlying framework of PEABS was made for being used with Java, Unity, and Apache Cordova, and it has several methods to create agents and and their behaviors. Thanks to Unity and PEABS, we have built a suitable environment for simulating strategies of attacks and defenses.

We selected PEABS instead of other agent-oriented methodologies because it combined short development time, technological support for software development, and high performance of the resulting systems in the particular case of ABSs. For instance, other theoretical methodologies such as the Gaia agent-oriented methodology lacked technological support for development, and other practical methodologies like Ingenias generated less efficient systems. In addition, we used the framework of PEABS instead of other well-known agent-oriented frameworks such as the Java Agent Development Framework (JADE), because PEABS allowed one to develop more efficient systems in the specific case of ABSs.

The definition of strategies is different between attack and defense strategies. For defining an attack strategy, users must create a new class that inherits from "MalwareAgent" class. This class must overwrite the Live method. Users can define fields for storing or analyzing any information. The Live method can call "AskService" to simulate the requests of services. The objects of this class should coordinate to ask services simultaneously in specific simulation iterations to achieve that the service is denied to honest agents.

To define a new defense strategy, users implement a new class that inherits from "ServerAgent". This class should overwrite the "DecideWhetherToProvideService" method for defining the reactive behaviors. It can also overwrite the Live if it needs to take any proactive action per iteration. The reactive behavior occurs when the strategy must react to certain event. In this case, the strategy reacts at the moment when a customer asks for a service. The implementation of the DecideWhetherToProvideService is normally the core of the new defense strategy. This method should decide whether to provide the service to this sender, by only knowing which is identifier. The strategy can use new class fields to store and analyze the history of requests of each agent ID. The implementation of the Live method is useful for performing any operation that needs to be taken only once per iteration or is related to the analysis of the collective behavior.

2.2. Strategies Defined with ABS-DDoS. Following the current approach with ABS-DDoS, we have defined the attacking strategies: (a) Coordinated Fixed Interval Attacker Agent, (b) Half-and-Half Attack, and (c) Substitute Attack. In addition, we have defined the defense strategies Frequency Defense and Coordinate Defense Server Agent.

2.2.1. Coordinated Fixed Interval Attack. In the strategy Coordinated Fixed Interval Attack (CFIA), all the attacker agents are coordinated to attack together periodically in a shared fixed interval.

As one can see in the diagram of Figure 3, all malware agents are waiting for an attack moment. In each step, they check the iteration number representing the time stamp. If

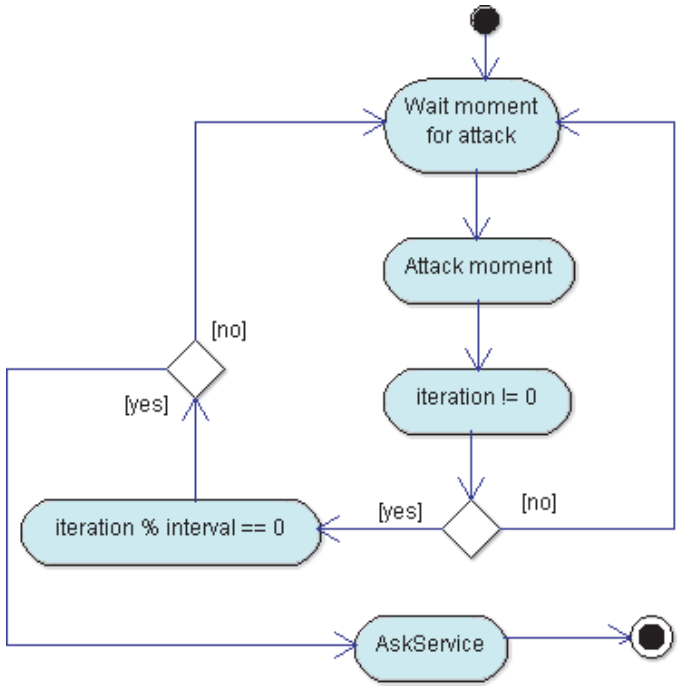


Figure 3: Coordinated fixed interval strategy.

the iteration is the first one, they will do nothing and keep waiting. If malware agents are not on their first iteration, they will check the attack moment. The attack moment is based on a fixed interval of iterations, and this interval can be set up by the user. For instance, if the simulation has 25 iterations in whole and user has set the interval to 5, then malware agents will attack 5 times during the whole simulation. Finally, the malware agents calculate this moment by calculating the remainder of dividing the iteration number and fixed interval. If the result is 0, malware agents will attack. Otherwise, they will keep waiting for another attack moment.

2.2.2. Half-and-Half Attack. The Half-and-Half (HaH) strategy is aimed at making its behavior more difficult to be detected than CFIA. Half-and-half attack is a natural evolution of CFIA. In this strategy, only half of the malware agents perform the requests in a certain iteration, and the other remaining half perform the requests in the other attack moments. In the diagram of Figure 4, one can see how HaH strategy works. Like in CFIA server agent is waiting for the attack moment and then it checks whether it is not in first iteration and it is in a suitable iteration (it means the iteration must match with the fixed interval set up by the user). Then, the agent decides whether it should attack. For this purpose, the agent determines whether the current iteration number is even or odd. Each agent has an integer ID number. In each even iteration, the strategy provokes that malware agents with even ID number request services to the target server agent. In each odd iteration, malware agents with odd ID number will send requests. Finally, when each malware agent knows if it is in even or odd iteration, it asks service. For instance, if our simulation has 25 iterations and 100 malware agents, in iteration number 5, 50 malware agents will attack server

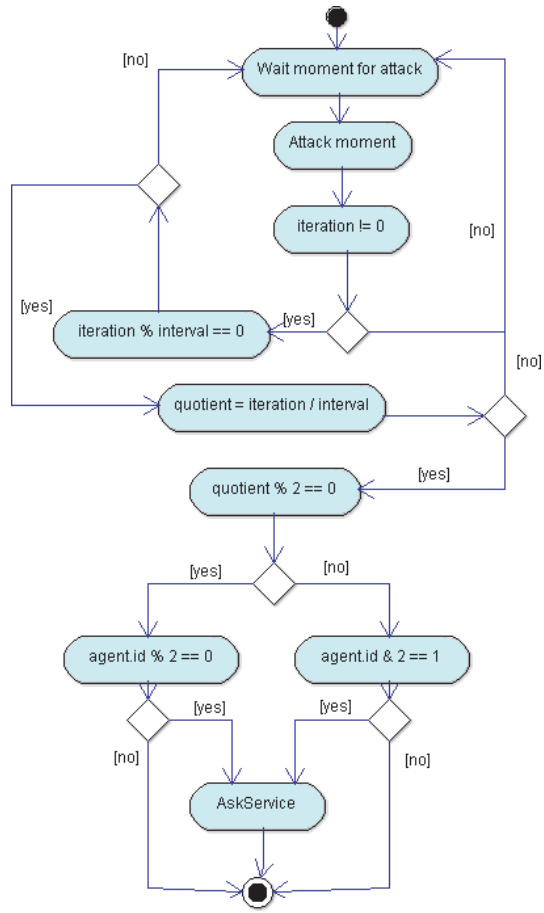


Figure 4: Strategy half and half.

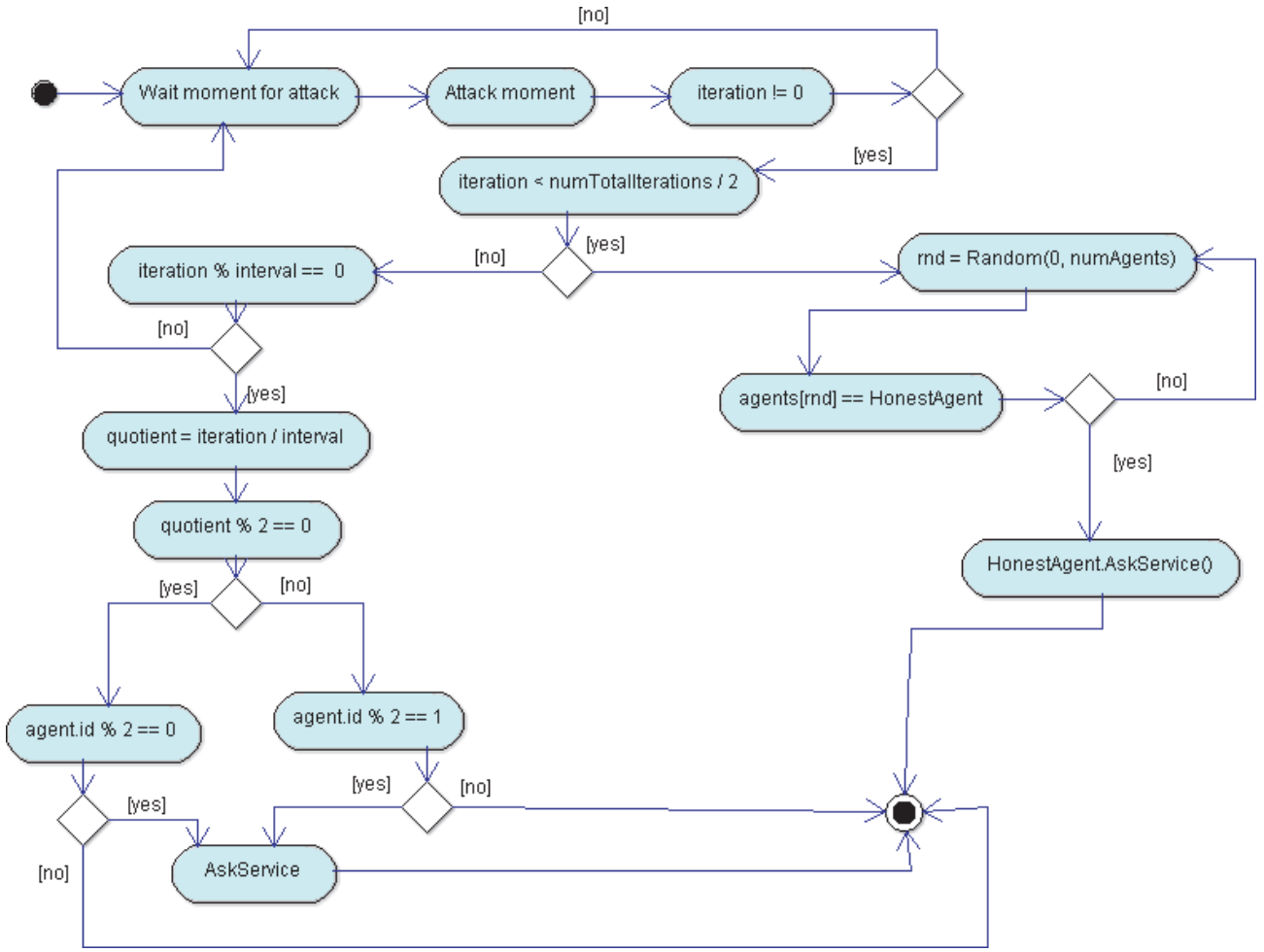


Figure 5: Substitute attack strategy.

agent. The next attack it will be in iteration number 10, but they will be the other remaining 50 malware agents.

2.2.3. Substitute Attack. Substitute Attack (SA) simulates the interception of encrypted messages and forwarding these. The purpose is to overload the target server agent, although these messages of requests may not be necessarily addressed to the target agents. The advantage of this attack is that the few successful forwarded messages that can actually be redirected to the same service use the identifier of a different agent. Thus, these are more difficult to be tracked. This attack would use the well-known man-in-the-middle attack [14] to acquire these message.

This attack follows two phases. The functionality can be seen in the diagram in Figure 5. In the first phase, this chooses a random agent from the ones that have requested the service. Then, it checks if the agent selected is an honest agent (by checking whether it is not one of the fellow attacker agents). If selected agent is not honest agent, the malware agent chooses other agents randomly until he catches an honest agent. When an honest agent is selected, a malware agent asks a service as if it was this honest agent. In this way, malware agents may saturate a server agent faster and the latter may deny real service requests in the next iterations.

On the other hand, the second phase of this strategy occurs in simulation's second half. In this second phase, malware agents attack normally; it means that they execute their own method "AskService". Moreover, in the second phase malware agents use HaH strategy in the same way as it was explained in the previous section.

2.2.4. Frequency Defense. Frequency Defense (FD) has like a main aim detecting agent's frequency on asked a service. If FD detects a high frequency it will not give service a certain agent. In order to determine whether an agent has a high frequency, FD measures the percentage of iterations in which it requests a service. If this agent requests services over a certain threshold, FD will deny to give a service. The aforementioned threshold is defined as an internal parameter. When this agent is created, the FD creates an index about the number of requests from each requester agent (unknown either malware or honest agent). In this way, Server Agent can save the absolute frequency of each agent. Then this frequency is divided by the number of iterations to obtain a relative frequency.

The reactive behavior of defense strategies is defined in "DecideWhetherToProvideService". In this method, a defense strategy decides whether to provide service regarding

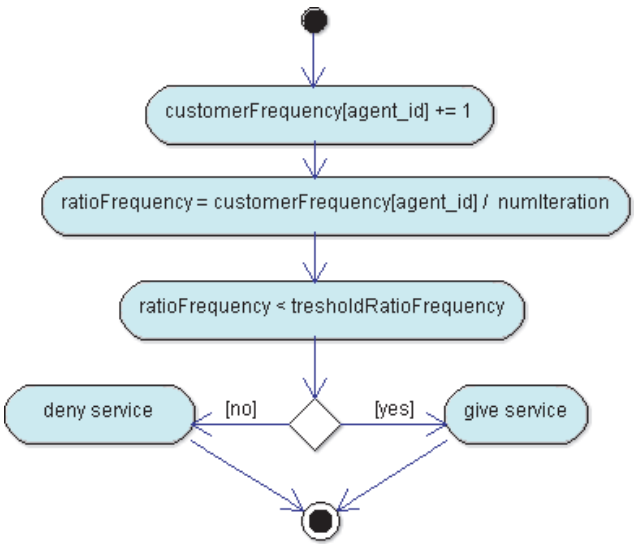


Figure 6: Strategy frequency server agent.

its estimation about whether the request was made by a malware agent.

Figure 6 depicts the process about how FD decides whether to provide a service or not. Each time a server agent receives a request, it updates its frequency record about the sender identified by its ID, including its absolute frequency and its frequency per time unit. The latter frequency is calculated as follows:

$$f_t(x) = \frac{f(x)}{(I_n + 1)} \quad (1)$$

where $f_t(x)$ is the frequency of requests of agent x per time unity, $f(x)$ determines the absolute frequency of agent x, and $I_n + 1$ determines the number of iteration (representing the number of hours simulated up to current state of the simulation).

Finally, if the frequency pert time unit is lower than a parametric internal threshold, Server Agent will provide service to this agent. Conversely, if this frequency is greater than the threshold, Server Agent will not provide service.

2.2.5. Coordinated Defense. The main goal of Coordinate Defense (CD) is to detect when it has been attacked by strategies with patterns similar to CFIA. Since this strategy is more complex than FD, we are going to explain it in three phases: Constructor Phase, Decide Phase, and Perform Phase. Constructor Phase occurs when the simulator runs CD's constructor. In constructor phase, CD sets up all initial parameters and initial variables. On the one hand we have two vectors, one of them will count all requests made by agents in only one iteration, and the other of them will count all requests made by agents in whole simulation. A request threshold is assigned to a product of the maximum amount of services given in one iteration and the ratio of frequency threshold. Both variables are established with arbitrary value given by programmer. The request threshold is defined with the following equation:

$$t = S_{max} * R \quad (2)$$

where t is request threshold, S_{max} is the max number of services that can be provided per iteration, and R is the ratio threshold request.

In addition, it initializes a counter for counting the number of requests the server agent. In this strategy, if the number of requests surpasses the requests thresholds, the server agent will assume that it is being attacked.

The second phase, Decide Phase, occurs when CD has to decide whether to provide a service. Figure 7 shows the flow diagram of this phase. In this diagram, one can see all the process inside the overwritten method "DecideWhether-ToProvideService". Each time an agent asks a service, CD counts it as like as FD and then increases the variable that represents the number of requests in an iteration. Until this point is similar to FD, from this point the process changes. CD checks whether it has been attacked. If it has not been attacked, it gives service. If CD has been attacked previously, CD calculates the ratio of requests in attacks. It is calculated as quotient between (a) the amount of requests that a certain agent has made in all the requests in iterations identified as attacks and (b) the number of these iterations identified as attacks. Finally, ratio request in attack is compared with threshold frequency in attack. The threshold frequency in attack is an internal parameter different from the one previously presented. It represents the barrier for discriminating the estimation between real requests and fake ones, based on how frequently an agent performs requests when DDoS attacks are detected. If the ratio request in attack of an agent surpasses this threshold, CD denies the service to this agent.

The final phase, the Perform Phase, occurs when CD agent has its turn for performing proactive actions. This is implemented overriding the "Live" method as instructed by PEABS. When the simulator finishes creating all agents, then it commands all agents to execute their own inherited method "Live". The first agents that execute this method are malware agents and honest agents. Then, CD executes this method. In method "Live". CD compares the amount of requests that has received only in current iteration with the threshold of requests. If the amount of requests is greater than the threshold of requests, it increases the total counter of attacks ("numAttacksTotal" in the diagram of Figure 7). In this case, it also updates index about the amount of times that each agent asks a service in the iterations identified as attacks (referred as "totalRequestInAttack in the diagram), using the record about the number of times each agent asked a service during the current iteration (denoted as "currentRequest"). Finally, in all the iterations regardless whether an attack was detected, the currentRequest is reset to zero times for each agent; the counter of the requests in the current iteration (referred as "numRequestInAIteration") is also reset to zero.

2.3. Method of Conducting the Experiments. We were alternatively defining attack and defense strategies. Each attack was aimed at exploiting the vulnerabilities of the previous defense. Each defense was aimed at protecting from the previous attack.

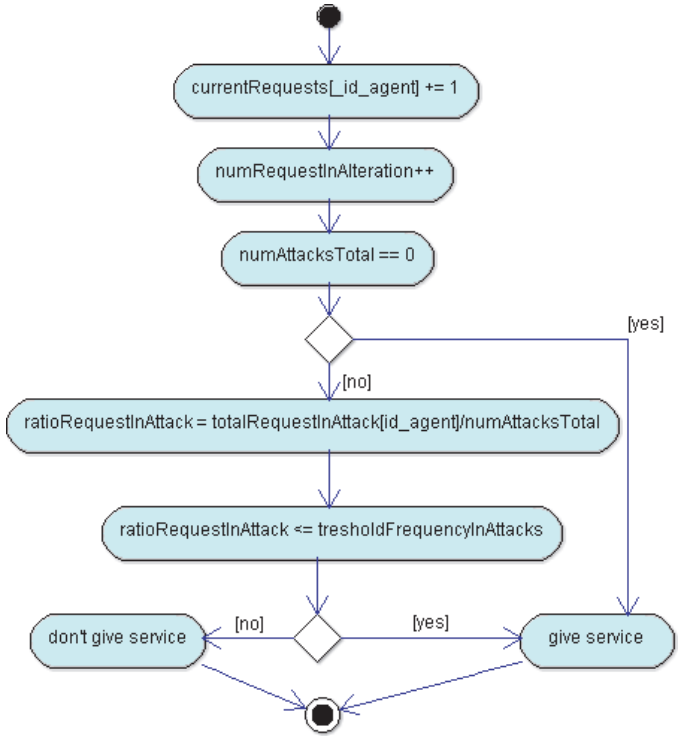


Figure 7: Strategy Coordinate Defense Server Agent.

Table 1: Simulated combinations of attack and defense strategies.

Defense Strategy	*Attack Strategy*
Frequency Defense	Half and Half Attack
Frequency Defense	Substitute Attack
Frequency Defense	Coordinated Fixed Interval Attack
Coordinated Defense	Half and Half Attack
Coordinated Defense	Substitute Attack
Coordinated Defense	Coordinated Fixed Interval Attack

The first simple attack was a continuous requester. The first defense was FD. From this point forward, we developed the strategies that we introduced in Section 2.2.

Then, we analyzed all the possible combinations of defense and attack strategies from the ones described in Section 2.2. Table 1 shows the combinations that we tested.

For the observation, we analyzed a short interval of the simulated time (25 h), to understand the periodic behavior. We also analyzed long interval to observe the evolution in the long term (100 h). In all the experiments, we used 100 malware agents and 30 honest agents.

In order to assess the performance of the most advanced defense strategy, we executed each with the most advanced attack strategy (i.e., SA). We performed several tests increasing, respectively, the number of agents performing requests and the simulated time.

3. Results and Discussion

Figures 8 and 9 show evolution results of simulating CFIA attacks on an agent defending with FD strategy. FD has

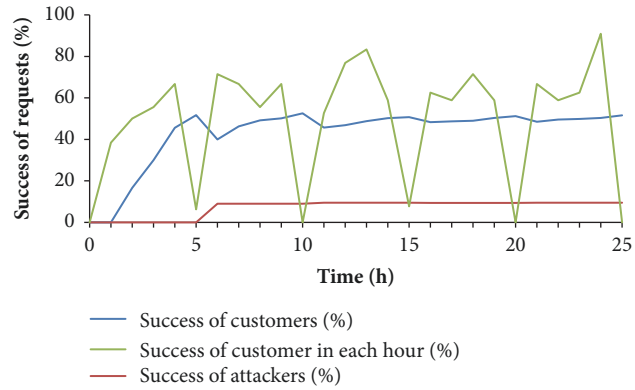


Figure 8: FD versus CFIA, simulation evolution of 25 h.

a vulnerability that CFIA can exploit. CFIA can reduce its frequency of requests by decreasing its frequency of coordinated attacks. In this way CFIA agents may not be detected by FD strategy.

The success of attackers is low, probably because the number of attackers is much higher than the number of possible services per iterations. Thus, only a few attackers get the service.

CFIA strategy achieve a successful DDoS, because customers only succeeded 50% in average. In addition, the results of the customer success per hour is the most evident fact that the DDoS attacks succeeded, since this measure indirectly revealed the proportion of denials to real service requests. The customer success per hour decreased to zero or almost zero in the specific hours where DDoS was executed.

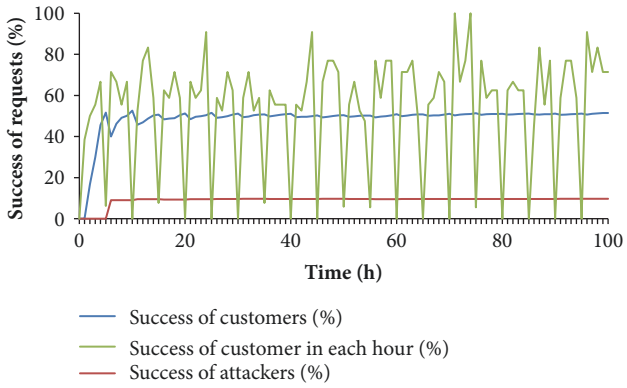


Figure 9: FD versus CFIA, simulation evolution of 100 h.

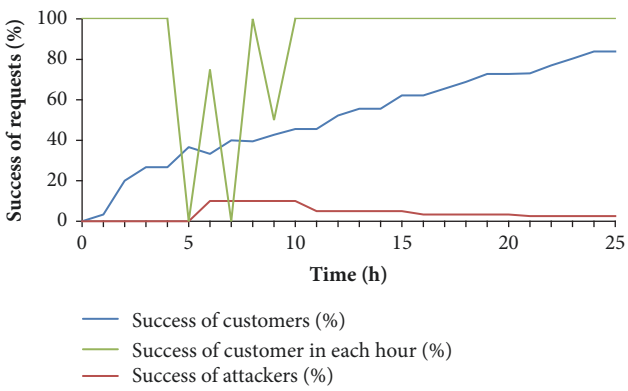


Figure 10: CD versus CFIA, simulation evolution of 25 h.

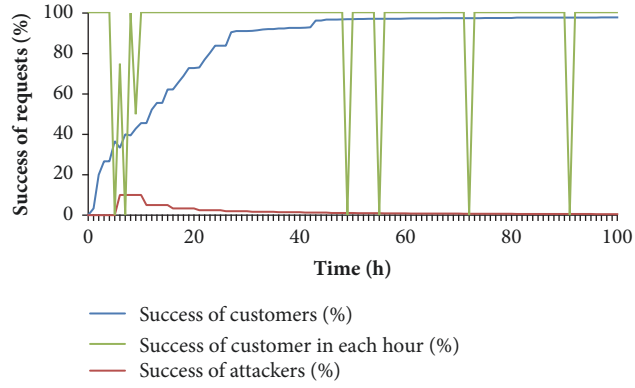


Figure 11: CD versus CFIA, simulation evolution of 100 h.

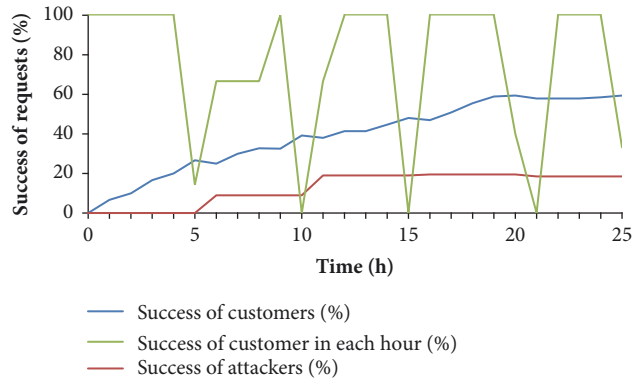


Figure 12: CD versus HaH, simulation evolution of 25 h.

The beginning of the global success of customers started with low a value because of the first attack was conducted in the first iteration. This value increased when this average measure was compensated in the in-between iterations without attacks.

The success of attackers is low because there are much more attackers than services the server agent can provide. However, the relevant measure of DDoS success is to achieve the fact that the service is denied to most customers.

Figures 10 and 11 show evolution examples of simulating CFIA attacks to CD strategy.

In fifth hour (the first attack), the attackers were not detected because they have not been tracked, and moreover, it is the first attack. Therefore, someone of them had success (10%). As the simulation progressed, in the next attacks, the attackers were well tracked; therefore the percentage success is going down near 0 after simulating 100 h.

Some of the honest agents were misclassified as attackers, and therefore at the next iterations they were discriminated. For this reason, one can see the percentage of customer success going down. In the second attack (after simulating 10 h), we can see that the relative amount of misclassified honest agents decreased. It is improbable that two honest agents asked a service in two different moments of attack. Specifically, the probability that an agent asks a service is 10%. Therefore, the probability of the same agent asks a service in the two first attacks is 0.01 (0.1 x 0.1), and the probability of asking in the three first attacks is 0.001 (0.1^3) and so on.

After simulating 45 h, sometimes the customer success per hour decreased to zero. The reason is probably that in these interactions none honest agent performed a request, and then the measure outputted the zero default value when avoiding raising the exception of division by zero.

Figures 12 and 13 show the results of performing attacks following HaH strategy to the defense following CD strategy. One can observe that HaH is more difficult to be tracked than CFIA, since in this strategy the attacks are not always performed by the same agents. The DDoS were successful since in most attacks, the customer success per hour reduced to zero. In addition, the global success of attackers was relatively high (around 20%). Since only the half of the attackers were used, a lower number of requests were denied for both honest and malware agents.

Figures 14 and 15 show the simulation of the combination of HaH attack strategy and FD defense strategy. FD defense is much more flexible in detecting attacks by frequencies, as it considers all the iterations and not only the ones suffering attacks. Hence, all the DDoS attacks successfully achieved the denials of services to honest agents, as one can observe in the decreases to zero or near-to-zero values in the attack iterations.

Figures 16 and 17 show simulation evolution examples of SA attacks and FD defenses, while Figures 18 and 19 show

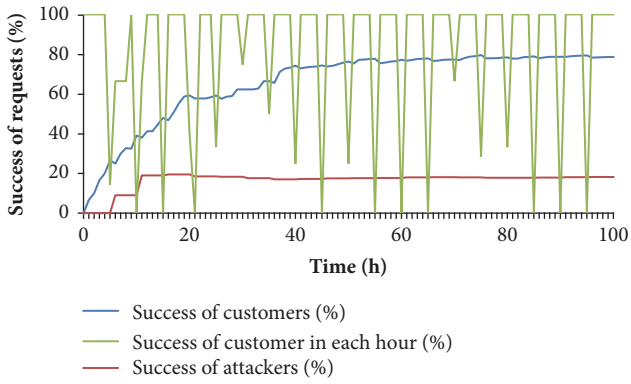


Figure 13: CD versus HaH, simulation evolution of 100 h.

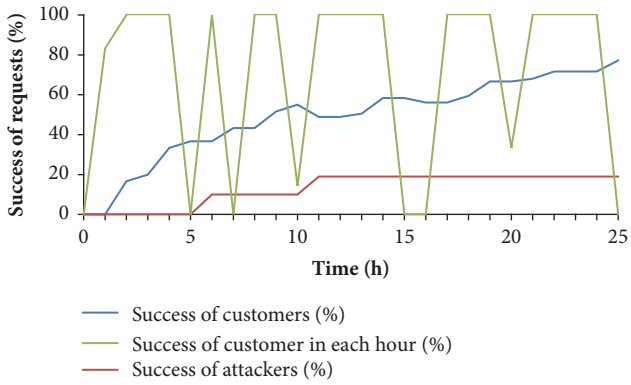


Figure 14: FD versus HaH, simulation evolution of 25 h.

simulation evolutions of SA attacks and CD defenses. SA was effective for exploiting the vulnerabilities of both defense strategies. The reason is that the impersonation of some honest agents make them look like attackers to both defense strategies. Thus, the success of customers per hour decreased not only in the attacked iterations but also in the others.

In Figure 19, in the middle of the simulation (i.e., about 50 h of simulation), one can observe a different behavior in the success of customers. In the second phase the success of customers do not depend on the possibility of being able to apply man-in-the-middle.

Although, SA seems to be the most effective attack theoretically, it depends on the ability of finding and impersonating other honest agents requesting a specific service.

Figures 20 and 21 show the response times in deciding whether to provide service for each request of the CD defense strategy when trying to defend from SA attack. One can observe that the average response time does not increase when augmenting the simulation duration. Thus, this defense strategy could probably run continuously without losing defense. By contrast, the time response for deciding whether to provide service increases when increasing the number of agents performing requests to a given service. However, the response times had a low absolute value. In addition, both FD and CD have a constant computational costs since the records and indexes can be accessed and updated in constant computational cost. Therefore, the current approach

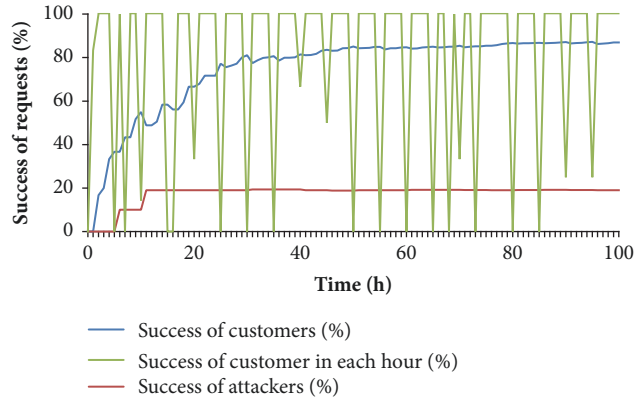


Figure 15: FD versus HaH, simulation evolution of 100 h.

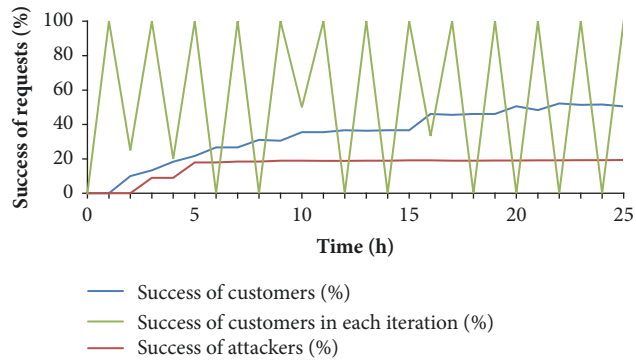


Figure 16: FD versus SA, simulation evolution of 25 h.

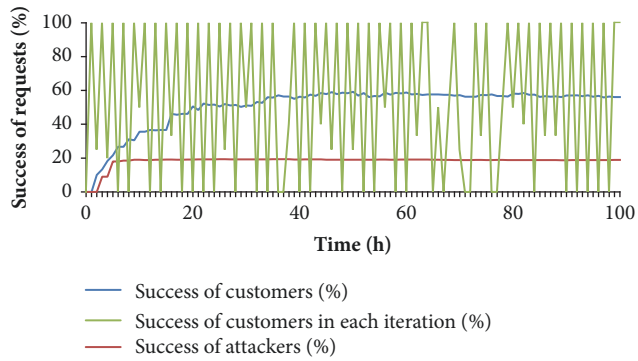


Figure 17: FD versus SA, simulation evolution of 100 h.

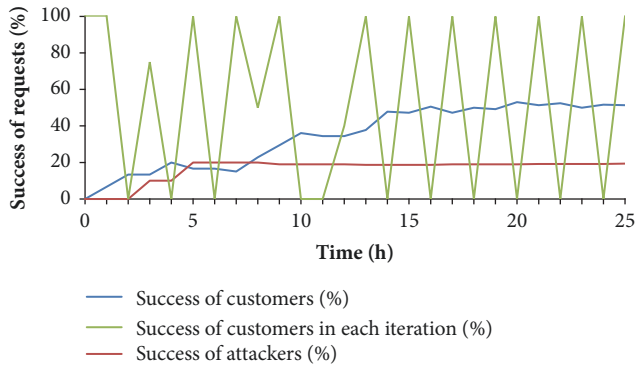


Figure 18: CD versus SA, simulation evolution of 25 h.

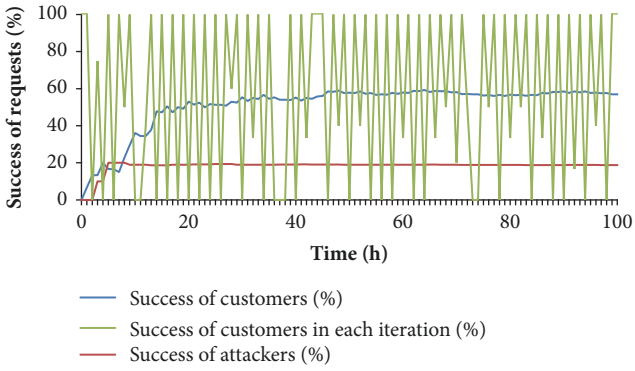


Figure 19: CD versus SA, simulation evolution of 100 h.

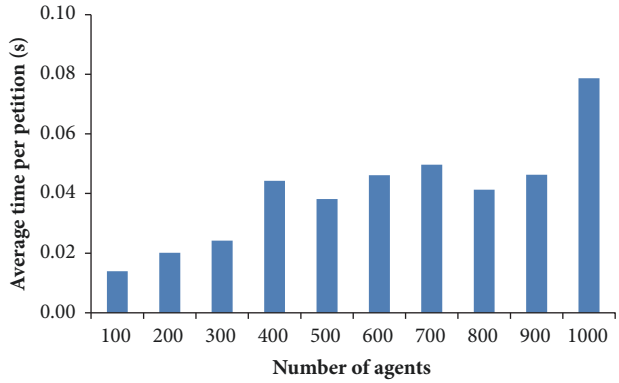


Figure 20: CD versus SA, response times when increasing the number of agents.

is efficient enough to be applied in SN sensors and IoT devices for performing efficient data forwarding or other services.

4. Conclusion and Future Work

The current work has presented a mechanism for improving security concerning DDoS attacks, by allowing developers to easily define and assess both attack and defense strategies in this context. The current approach also allows detecting DDoS security challenges by defining DDoS attack strategies that are usually to track for being counteracted. The current approach is based on the novel ABS called ABS-DDoS. We have defined and assessed two defense strategies and three attack strategies with ABS-DDoS. This ABS helped us to understand the results of all the possible combinations. In addition, we defined defense strategies that were efficient in terms of time response for deciding whether to provide services. In this way, these strategies can be used for maintaining security in SN sensors and IoT devices with low processing capabilities from DDoS attacks.

The proposed ABS is planned to be extended in order to simulate other types of attacks such a man-in-the-middle and zero-day attack. This may require define new agent types that will allow defining defense and attack strategies for attacks like man-in-the-middle attack or zero-day attack. These agent types would need to incorporate and manage the necessary

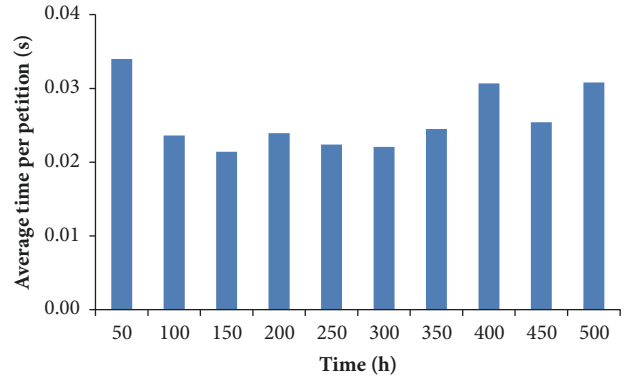


Figure 21: CD versus SA, response times when increasing the number of simulated hours.

information for measuring the effectiveness of defense and attack strategies in the context of each security attack. For example, in the case of the man-in-the-middle, a metric could be the percentage of messages that have successfully intercepted and forwarded.

Moreover, we plan to deploy advanced defense and attack strategies in real scenarios. We plan to test some attack strategies in SNs and IoT devices to exploit their vulnerabilities. Then, we will install defense strategies in SN sensors and IoT devices to protect from these attacks, in order to measure response times and effectiveness of the defense strategies in these devices with low processing capabilities. We also plan to test defense and attack in cloud services, which are one of the most frequent target nowadays, according to [15]. If we have the chance, we can test in important websites such as the ones from any government or big company. Finally, we could organize security contexts in which participants define defense and attack strategies with ABS-DDoS and compete against each other by means of the simulator.

Data Availability

All the relevant data of the current work are mentioned in the article or shown in its graphs.

Conflicts of Interest

The authors declare that there are not any conflicts of interest about the current work.

Acknowledgments

The authors acknowledge the research project "Construcción de un Framework para Agilizar el Desarrollo de Aplicaciones Móviles en el Ámbito de la Salud" funded by University of Zaragoza and Foundation Ibercaja with Grant Reference JIUZ-2017-TEC-03. This work has been supported by the program "Estancias de Movilidad en el Extranjero José Castillejo para Jóvenes Doctores" funded by the Spanish Ministry of Education, Culture and Sport with Reference CAS17/00005. The authors also acknowledge support from

"Universidad de Zaragoza", "Fundación Bancaria Ibercaja", and "Fundación CAI" in the "Programa Ibercaja-CAI de Estancias de Investigación" with Reference IT1/18. This work acknowledges the research project "Desarrollo Colaborativo de Soluciones AAL" with reference TIN2014-57028-R funded by the Spanish Ministry of Economy and Competitiveness. It has also been supported by "Organismo Autónomo Programas Educativos Europeos" with Reference 2013-1-CZ1-GRU06-14277. Furthermore, they acknowledge the "Fondo Social Europeo" and the "Departamento de Tecnología y Universidad del Gobierno de Aragón" for their joint support with Grant no. Ref-T81.

References

[1] H. M. Jawad, R. Nordin, S. K. Gharghan, A. M. Jawad, and M. Ismail, "Energy-efficient wireless sensor networks for precision agriculture: A review," *Sensors*, vol. 17, no. 8, 2017.

[2] I. García-Magariño, R. Lacuesta, and J. Lloret, "ABS-FishCount: An Agent-Based Simulator of Underwater Sensors for Measuring the Amount of Fish," *Sensors*, vol. 17, no. 11, p. 2606, 2017.

[3] S. H. Lee, S. Lee, H. Song, and H. S. Lee, "Wireless sensor network design for tactical military applications : Remote large-scale environments," in *Proceedings of the MILCOM 2009 - 2009 IEEE Military Communications Conference*, pp. 1–7, Boston, MA, USA, October 2009.

[4] I. García-Magariño, R. Lacuesta, and J. Lloret, "Agent-Based Simulation of Smart Beds With Internet-of-Things for Exploring Big Data Analytics," *IEEE Access*, vol. 6, pp. 366–379, 2018.

[5] T. Anagnostopoulos, K. Kolomvatsos, C. Anagnostopoulos, A. Zaslavsky, and S. Hadjiefthymiades, "Assessing dynamic models for high priority waste collection in smart cities," *The Journal of Systems and Software*, vol. 110, pp. 178–192, 2015.

[6] X. Yang, S. Zhou, G. Ren, and Y. Liu, "Computer network attack and defense technology," *Information and Computer Security*, vol. 1, no. 1, pp. 35–41, 2018.

[7] E. Alomari, S. Manickam, B. B. Gupta, S. Karuppayah, and R. Alfaris, "Botnet-based distributed denial of service (DDoS) attacks on web servers: classification and art," *International Journal of Computer Applications*, vol. 49, no. 7, pp. 24–32, 2012.

[8] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "IoTPOT: analysing the rise of IoT compromises," *EMU*, vol. 9, pp. 1–9, 2015.

[9] Z. Anwar and A. W. Malik, "Can a DDoS attack meltdown my data center? A simulation study and defense strategies," *IEEE Communications Letters*, vol. 18, no. 7, pp. 1175–1178, 2014.

[10] S. Huda, R. Islam, J. Abawajy, J. Yearwood, M. M. Hassan, and G. Fortino, "A hybrid-multi filter-wrapper framework to identify run-time behaviour for fast malware detection," *Future Generation Computer Systems*, vol. 83, pp. 193–207, 2018.

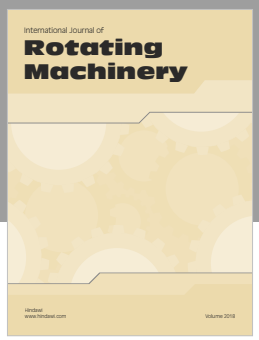
[11] C. Gkountis, M. Taha, J. Lloret, and G. Kambourakis, "Light-weight algorithm for protecting SDN controller against DDoS attacks," in *Proceedings of the 2017 10th IFIP Wireless and Mobile Networking Conference (WMNC)*, pp. 1–6, Valencia, September 2017.

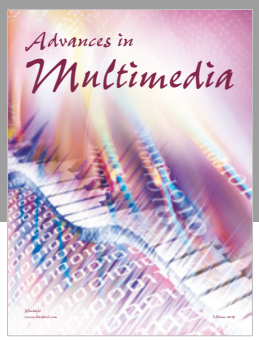
[12] I. García-Magariño, G. Palacios-Navarro, and R. Lacuesta, "TABSAOND: A technique for developing agent-based simulation apps and online tools with nondeterministic decisions," *Simulation Modelling Practice and Theory*, vol. 77, pp. 84–107, 2017.

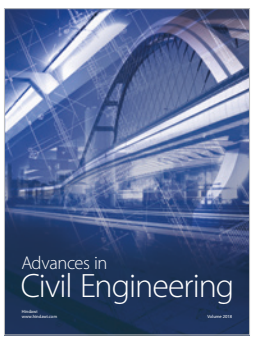
[13] I. García-Magariño, A. Gómez-Rodríguez, J. C. González-Moreno, and G. Palacios-Navarro, "PEABS: a process for developing efficient agent-based simulators," *Engineering Applications of Artificial Intelligence*, vol. 46, pp. 104–112, 2015.

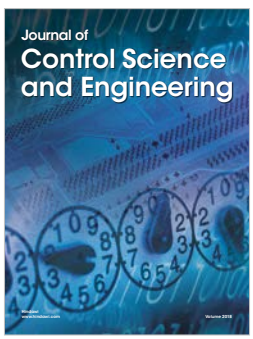
[14] A. Akhunzada, M. Sookhak, N. B. Anuar et al., "Man-At-The-End attacks: Analysis, taxonomy, human aspects, motivation and future directions," *Journal of Network and Computer Applications*, vol. 48, pp. 44–57, 2015.

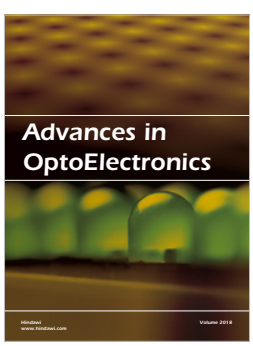
[15] Q. Yan, F. R. Yu, Q. X. Gong, and J. Q. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: a survey, some research issues, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 602–622, 2016.

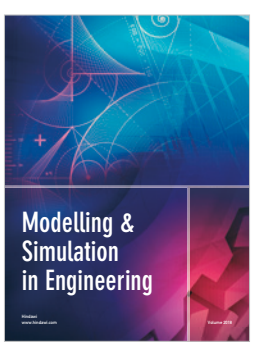


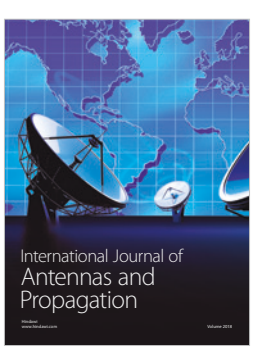


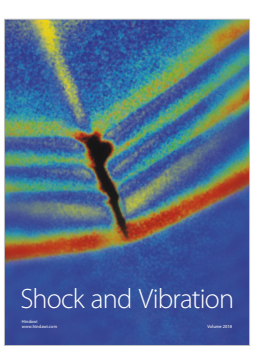


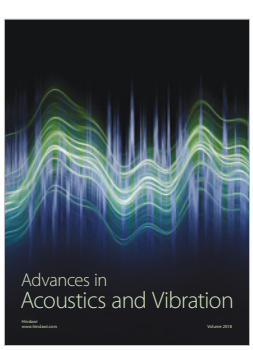












Hindawi

Submit your manuscripts at
www.hindawi.com