Hindawi
Security and Communication Networks
Volume 2017, Article ID 8575842, 9 pages
https://doi.org/10.1155/2017/8575842

WILEY | Hindawi

Research Article

ABS-TrustSDN: An Agent-Based Simulator of Trust Strategies in Software-Defined Networks

Iván García-Magariño[1,2] and Raquel Lacuesta[1,2]

[1]*Department of Computer Science and Engineering of Systems, Escuela Universitaria Politécnica de Teruel, University of Zaragoza, c/Atarazana 2, 44003 Teruel, Spain*
[2]*Instituto de Investigación Sanitaria Aragón, University of Zaragoza, Zaragoza, Spain*

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

Received 28 July 2017; Accepted 6 September 2017; Published 11 October 2017

Academic Editor: Huaizhi Li

Software-defined networks (SDNs) have become a mechanism to separate the control plane and the data plane in the communication in networks. SDNs involve several challenges around their security and their confidentiality. Ideally, SDNs should incorporate autonomous and adaptive systems for controlling the routing to be able to isolate network resources that may be malfunctioning or whose security has been compromised with malware. The current work introduces a novel agent-based framework that simulates SDN isolation protocols by means of trust and reputation models. This way, SDN programmers may estimate the repercussions of certain isolation protocols based on trust models before actually deploying the protocol into the network.

1. Introduction

Software-defined networks (SDNs) allow separating control and data planes, offering better network management rather than traditional networks. In SDNs, programming does not need to be performed node by node, but in a centralized way. This provides more flexible, efficient, and scalable networks. This way, the network is implemented independently of manufacturers or component models. The SDN controller is responsible for acting as a centralized control point at a logical level. The controller has the task of coordinating communications between applications that interact with network elements. However, this centralized control scenario introduces some security challenges. In order to protect a network, one should secure data, controllers and devices, and communications. In the literature, some works increased security in SDNs including authentication mechanisms, controller replication schemes, and policy conflict resolution schemas. Alterations in the network's components and in their behavior need to be assessed to sustain network security. That way, the network's components trust and reputation become an important issue to deal with.

Due to the global network view that the SDNs provide, we can introduce a detection system to assess the devices' reputation and trustworthiness. Based on the analysis, the controller can reprogram the network operation. In this scenario, the SDN architecture is used to improve the network security using a security monitor that analyzes anomaly-detection behaviors of the network's components. This approach can improve the robustness of the network to detect attacks.

Monitoring systems have been proposed by some authors in order to protect networks from attacks. Some of them use a centralized scheme. For example, OpenSAFE system [1] was proposed to enable the arbitrary direction of traffic for security monitoring applications. A flow specification language named ALARMs was used for arbitrary route management through monitoring devices. Braga et al. [2] presented a lightweight method for DDoS attack detection. In the proposal, they monitored NOX switches at regular intervals to identify abnormal flows. In [3], an Intrusion Detection-Prevention System architecture was proposed for the cloud virtual networking environment. It inherited the detection capability from Snort and the flexible network reconfiguration from SDNs. CloudWatchers, a framework

used in [4], monitored network flows for large and dynamic cloud networks.

Other authors presented distributed architecture models to distribute the security efforts among the network's devices. For instance, a distributed approach was presented in [5] where the control and trust were delegated to end hosts and users. Therefore, the network's devices participated in network security enforcement. In [6], Resonance, a dynamic access control system, was carried out by network devices. The system used programmable switches to manipulate traffic at lower layers, enforcing high-level security policies. A deep packet inspection module was proposed by Goodney et al. [7]. The proposed engine was used as a simple network intrusion detection system.

Trust and reputation models have been applied for achieving security in networks. Yan et al. [8] proposed a security and trust framework using fifth-generation (5G) wireless systems. That work proposed an adaptive trust evaluation for deploying certain trustworthy security services in both virtualized networks and SDNs. Michiardi and Molva [9] proposed a collaborative reputation mechanism for promoting the cooperation among network nodes, preventing selfish behaviors. Liu and Issarny [10] proposed a reputation mechanism for mobile ad hoc networks, in which agents shared the reputation about other agents. This system promoted honest recommendations, by including the notion of recommendation reputation. The reputations were also calculated about the recommendations of other agents.

In general, multiagent systems (MASs) have been considered a proper mechanism for implementing and simulating trust and reputation models. For instance, the Agent Reputation and Trust (ART) testbed [11] provided a framework for testing different policies about trust on agents (estimated from the direct interaction) and reputation of other agents (obtained from the recommendations of other agents). Their framework illustrated these concepts in the domain of appraisals of paintings. They organized competitions, in which each participant programmed an agent, and all the agents were executed together. Other works explored how to integrate different trust measures in MASs, like in the one by Rosaci et al. [12]. They introduced concepts such as the reliability about each reputation value. Furthermore, Jelenc et al. [13] proposed an agent-based mechanism for evaluating trust models. They used a testbed, but their mechanism separated the measurement of trust models from the repercussions of the decision-making mechanisms. In addition, Chen et al. [14] presented a MAS trust model for wireless sensor networks. In their system, they propagated encrypted reputation information about the different nodes. They tested their system under the bad mouthing attacks (spread of false reputation information), conflicting behavior attacks (an agent is malicious towards one node and honest to the others, for provoking conflicts about the reputation of certain nodes), and on-off attacks (a node behaves well and badly alternatively). They also presented some simulations with their models.

Agent-based simulators (ABSs) are a specific kind of MASs intended for making simulations and have been applied to implement trust testbed, like in the work of Kim [15] in the domain of supply networks. That work tested adaptive behaviors based on trust and explored emergent outcomes such as self-organizing processes and macrolevel systems behaviors.

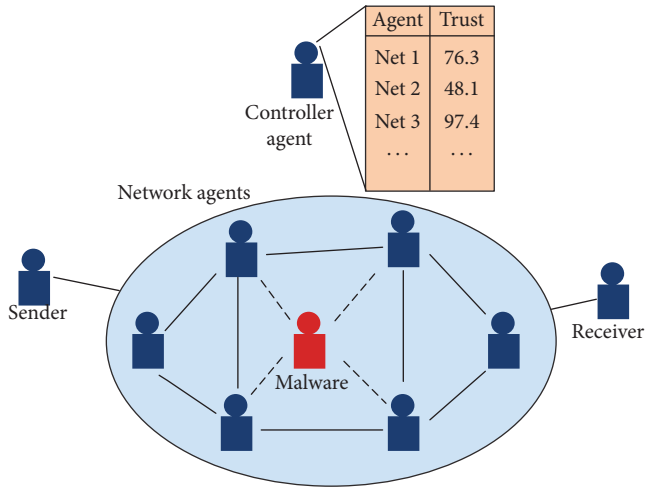


Figure 1: Overview of ABS-TrustSDN.

On the whole, the current literature shows the relevance of guaranteeing the security on SDNs. Trust models have been shown to be useful for this purpose. However, to the best of the authors' knowledge, the existing testbed for simulating trust strategies is not designed for simulating the repercussion of trust models specifically for SDNs. In this context, MASs have been shown to be useful for comparing and simulating trust models in several domains, especially when separating trust models from decision-making processes. ABSs may be a proper choice for developing trust testbeds and simulation frameworks.

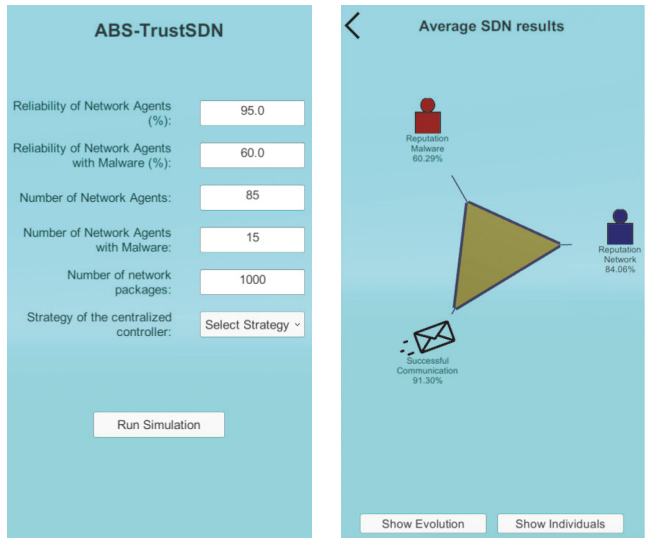
In this context, the current work presents an Agent-Based Simulator of Trust strategies in SDNs (called ABS-TrustSDN). In this simulator, an agent represents the centralized controller of the SDN, and this agent builds a trust model regarding the history of the network's components based on the network packets that were properly transmitted or lost through that component. Other agents represent the network's components that can either work properly or malfunction. The underlying framework allows designers to test different isolation protocols based on trust models. This work explores different strategies of the SDN controller based on trust with the presented approach and compares their outcomes.

2. Materials and Methods

2.1. ABS-TrustSDN

2.1.1. Overview. The current work uses the novel ABS-TrustSDN tool as the main material of this research. This tool and its underlying framework have been developed specifically for the current work, which is one of its main contributions. In order to guarantee the reproducibility of the experiments, this tool has been made available on its website [16].

This tool is an ABS in which agents simulate the behaviors of the SNDs' components. Figure 1 introduces graphically



(a) Inputs of the UI (b) Average results with a star plot

Figure 2: Main screens in the UI of ABS-TrustSDN.

an overview of this tool. The "network agents" represent the network's components such as groups of switches, and these can have different behaviors. Most of these work properly transmitting the network packages following the orders of the SDN centralized controller. However, a few of the network agents may have a malicious or nonproper behavior. This behavior can be intended for compromising the privacy and taking nonauthorized data. It can also be some malfunctioning that makes them lose a high rate of network packages. From this point forward, sometimes this kind of network agents will also be referred to as "malware agents" in order to distinguish them from all the other network agents that will be referred to as "normal network agents" in this context.

The "controller agent" represents the centralized controller of the SDN. It decides how to route the traffic of the network to avoid its saturation. In ABS-TrustSDN, this agent is mainly intended to incorporate the isolation protocol. This can be programmed by the SDN designer, in order to simulate several isolation protocols. The controller agent incorporates a table with its trust on each network agent as a percentage in the 0-to-100 interval.

The tool can simulate any number of messages from a sender to a receiver. In each simulated communication, the controller agent selects a route represented with several network agents. The receiver can report whether the message was properly received or whether it was lost. If lost, the controller agent can track in which point of the route the package was lost and decrease its trust in the corresponding network agent. Normally, if the transmission was correct, it slightly increases its trust on all the involved network agents.

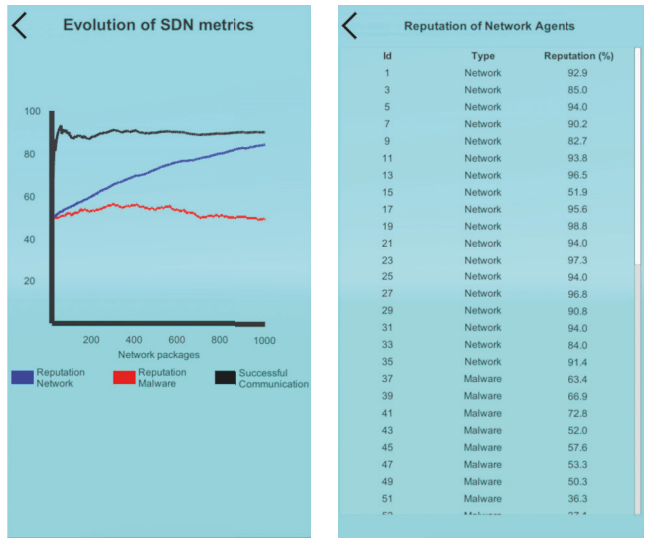
The network agents with a nonproper behavior only perform malicious or error-prone actions sometimes in a certain percentage of times, so these can be difficult to detect without an accurate isolation protocol. This percentage of misbehavior can be configured as an input of the simulator.

2.1.2. Definition of Strategies with ABS-TrustSDN. This tool was developed following PEABS (a process for developing agent-based simulators) [17]. It considered the common guidelines for designing proper agent-oriented architectures [18] and integrating these with web-based systems [19]. The communications were designed considering the common recommendations in the area of MASs [20]. This process was adapted to develop the app with the Unity engine and the C# language.

In each simulation iteration, the controller agent selects the next network agent to transmit from the available ones with a nondeterministic decision based on probabilities. These probabilities are determined by its internal trust model. This way, normally, this agent selects more frequently the agents that it trusts more according to its model.

The controller agent receives feedback about whether the transmission was correct or not for each transmitted package in each network agent. The framework allows extending the controller agent to define exactly the way of updating the trust by overwriting the method "Update Trust." This method receives the previous trust of the agent, the positive or negative result of the communication, and the ID of the network agent. It must return the new trust. Designers can implement basic behaviors regarding only the previous trust and the communication result. They can also implement more elaborated trust models with data structures inside the extended class for taking into account the previous histories of the agents identified by their IDs. Section 2.2 introduces several examples of strategies that were defined with the current approach.

2.1.3. User Interface of ABS-TrustSDN. Figure 2 shows the main screens of the user interface (UI) of ABS-TrustSDN. More concretely, Figure 2(a) shows the screen in which the user enters the input values of the simulations. These include the reliabilities of, respectively, the normal network agents



(a) Evolution of SDN metrics (%) (b) Reputation of network agents

Figure 3: Screens with the details of the simulation in the UI of ABS-TrustSDN.

and the ones with malware. Each reliability represents the percentage of times that a kind of network agents works properly. It is worth mentioning that even correct components can fail occasionally, and the network components with malware can work most of the times. Both aspects can be configured by the user with two percentages. The simulation can be set with different numbers of normal network agents and the ones with malware. The components with malware are usually a minority, but the user can simulate different scenarios about this.

The user can also set the number of transmitted network packages, to show both the short-term and the long-term repercussions of the different trust policies. Finally, the user can select one of the existing trust model strategies for the centralized controller. The user can easily define new strategies by using the underlying framework of ABS-TrustSDN, as previously introduced in Section 2.1.2. After entering the simulation inputs, the user can start running the simulation by pressing the corresponding "Run simulation" button.

After running the simulation, the ABS presents the average simulated outcomes with a star plot like the one in Figure 2(b). These outcomes include average reputations of, respectively, the normal network agents and the ones with malware. The star plot also includes the success rate of the transmitted network packages through the network agents in general. The user can also observe further details in the screens of the UI presented in Figure 3, by pressing any of the two bottom buttons in Figure 2(b).

The UI screen of Figure 3(a) shows the evolution of the average reputations of the two different kinds of network agents and the evolution of the success rate in the communications up to the corresponding time. The chart shows the evolution of these three variables evolving along the evolution time implicitly represented by the number of the network packages transmitted in the abscissa axis. In the UI screen of Figure 3(b), the user can see the reputations of all the network agents alongside their IDs and their abbreviated types (i.e., "Network" for the normal network agents and "Malware" for the ones with malware). The reputation of each network agent is determined by the percentage that the centralized controller agent trusts on it.

2.2. Experimental Method. The current work followed an exploratory design, as commonly done in works about ABSs [21] or more generally in MASs [22], in order to test the utility of the novel ABS-TrustSDN tool and its underlying framework. We implemented three different strategies of conforming trust models. Then, we experienced the strategies in two different scenarios represented with certain input configurations. Each combination of strategy and configuration was executed 1,000 times for avoiding bias due to the nondeterministic behavior of the simulator. The results were compared by presenting and discussing the average results. The results were analyzed with a statistical test to determine the significance of the differences. We calculated the effect sizes to measure these differences. Furthermore, we analyzed the evolutions of the different combinations in order to provide possible explanations about the resulting outcomes.

In this experimental method, we defined the following strategies with the underlying framework of ABS-TrustSDN:

(i) *Fixed Strategy*. This strategy just considers the last time the controller agent interacted with a certain network agent. If the communication was successful, then it assigns a high fixed trust. Otherwise, it assigns a low fixed trust. The high fixed trust is 100%, while the low trust is 10%. It does not assign a 0% trust, as a normal network agent may have had an uncommon failure. This way, the network agent may be contacted eventually in the future and then it will recover the high trust if it does not fail again.

Table 1: Input configurations.

	Configuration 1	Configuration 2
Reliability of network agents (%)	95.0	95.0
Reliability of network agents with malware (%)	60.0	15.0
Number of network agents	85	85
Number of network agents with malware	15	15
Number of network packages	1000	1000

Table 2: Average results of the 1,000 simulations for each pair of strategies and input configuration with SDs between parentheses.

Strategy	Configuration 1			Configuration 2		
	Reputation network (%)	Reputation malware (%)	Successful communication (%)	Reputation network (%)	Reputation malware (%)	Successful communication (%)
Fixed	73.55 (4.63)	22.88 (8.04)	92.60 (0.76)	73.48 (4.46)	11.55 (3.04)	91.88 (0.81)
Tabsaond	84.51 (0.75)	55.18 (3.56)	90.84 (0.86)	85.08 (0.74)	28.32 (2.22)	88.18 (0.92)
History	99.52 (0.68)	39.94 (13.52)	91.12 (0.76)	99.53 (0.67)	3.70 (4.76)	89.03 (0.74)

(ii) *Tabsaond Strategy*. If the communication was correct with a network agent, the controller increases the trust on it. Otherwise, it decreases the trust. It updates the trust considering a ratio of the distance to the approaching limit, following the recommendations of TABSAOND (a technique for developing agent-based simulation apps and online tools with nondeterministic decisions) [23]. The controller agent decision can be either to increase its trust ($d = 1$), decrease its trust ($d = -1$), or keep the trust with a neutral trend ($d = 0$). In particular, the controller agent uses the following formula to calculate the variation for altering the trust on a network agent:

$$v = \begin{cases} K * (U_L - x), & \text{if } d > 0, \\ -K * (x - L_L), & \text{if } d < 0, \\ 0, & \text{otherwise,} \end{cases} \quad (1)$$

where x is the current trust on this agent, K represents the ratio to the approaching limit (in this work $K = 0.15$), and L_L and U_L, respectively, represent the lower and the upper limits (i.e., 0 and 100, as the trust is represented with percentages).

(iii) *History Strategy*. This strategy assigns a high trust value to all the network agents by default. In each interaction with a specific agent, it records the communication result. It does not change the trust until it has analyzed a minimum number of interactions that is established as an internal parameter named "analysis window." In this experiment, this parameter was set to 5 interactions. Once this limit is reached, this strategy calculates the ratio of successful transmissions with the corresponding agent. If this ratio is below a certain threshold (in this experiments set to 60%), then the agent is set to a low trust for isolating the agent. In this case, the low trust was set to zero for the complete isolation, as it has made sure that its behavior is wrong for at least a certain number of interactions.

In order to experience the aforementioned strategies in different contexts, this work established two settings of the input parameters, which are referred to as input configurations. Table 1 shows the input values of these configurations. The difference between these two is that the rates of failures of the malware agents are different. This way, the strategies will be tested with different kinds of network agents with malware, represented with different reliability percentages (i.e., 60% versus 15%).

3. Results and Discussions

3.1. Comparison of the Final Simulated Outcomes of the Trust Strategies. As mentioned in the experimental method, the simulator was executed 1,000 times for each combination of strategy and input configuration. Table 2 presents the average results of these simulations with the SD between parentheses.

By observing the averages, one can extract several conclusions. First, when the input configuration varies considering different reliabilities of network agents with malware, the relative order of the strategies varies in the malware agents' reputation dependent variable. Thus, different strategies may be considered appropriate given the nature of the most expected kind of malware.

In addition, one can observe the results of the isolation of malware resources by observing the reputation of normal network agents and the reputation of the ones that got malware. An ideal strategy would trust network agents with 100% and would not rely on the ones with malware with trust of 0%. One can observe that History strategy is the one that gets closer to the ideal trust for isolating malware resources. When the agents with malware have a very low reliability

Table 3: Results of Welch's unequal variances t-test generalized for three samples. [a]Asymptotically F distributed. [**]Significant with a significance level of .001.

	Configuration 1			Configuration 2		
	Reputation network (%)	Reputation malware (%)	Successful communication (%)	Reputation network (%)	Reputation malware (%)	Successful communication (%)
Statistic[a]	119441.343	6993.621	1461.395	113571.231	17614.808	5368.600
df1	2	2	2	2	2	2
df2	1784.344	1613.377	1992.052	1783.826	1856.863	1983.274
Sig.	.000[**]	.000[**]	.000[**]	.000[**]	.000[**]	.000[**]

Table 4: Cohen's d effect sizes.

Configuration		Tabsaond			History		
		Reputation network (%)	Reputation malware (%)	Successful communication (%)	Reputation network (%)	Reputation malware (%)	Successful communication (%)
Fixed	1	2.34	3.67	−1.54	5.55	1.08	−1.39
	2	2.56	4.60	−3.02	5.77	−1.39	−2.59
Tabsaond	1				14.90	−1.09	0.24
	2				14.43	−4.78	0.72

(i.e., configuration 2), the history strategy assigns the highest reputation to network agents (i.e., 99.5%) compared to the other strategies and the lowest reputation to the agents with malware (i.e., 3.7%). In configuration 1 with malware agents with a higher reliability, Fixed strategy isolates them with a lower reputation, but it also isolates wrongly some normal network agents. Thus, even in this configuration, History strategy may be the most appropriate one for separating the trust values of network agents and malware agents.

Another relevant aspect is to analyze which strategy achieves the highest rate of successful communications. In this case, the Fixed strategy is the one that obtains the highest rate in both configurations. Notice that the rate of success in communications not only depends on the quality of the final trust model of the isolation protocol of malware resources, but also depends on how quick the isolation is taken. Slow isolation may cause the failure of many initial communications, and this will hinder the final communication rate. This may be the reason why the Fixed strategy got the highest rate, as it can start the isolation from the very first interaction with an agent.

The results were analyzed with Welch's unequal variances t-test generalized for more than two samples [24], in order to determine whether the differences were statistically significant. Table 3 shows the results of this statistical test. This test was selected as it is robust for comparing samples with unequal variances. The differences of the results between the different strategies were very significant with a significance level of .001 for all the dependent variables.

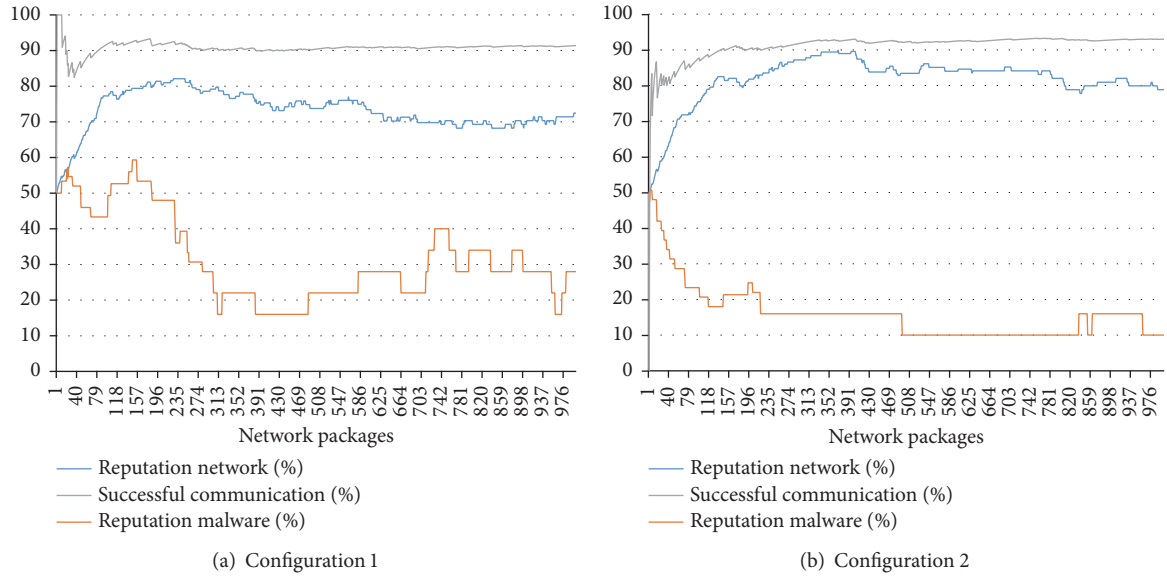
Table 4 analyzes Cohen's d effect sizes between each pair of strategies for each input configuration and dependent variable. Cohen's [25] guideline assigned .2, .5, and .8, respectively, to small, medium, and large effect sizes. According to this guideline, all the effect sizes of the current experiments were large between all the pairs of strategies and dependent variables, except in the case of the comparison of success rate in communications between Tabsaond and History strategies. In that case, the effect size was small and medium for, respectively, configurations 1 and 2.

On the whole, one can observe that ABS-TrustSDN has allowed comparing the repercussions of three different trust strategies on SDNs concerning two different kinds of attacks. These attack kinds were represented with their different ratios of communication failures implicitly indicated in the input configurations. The results presented significant differences. Hence, ABS-TrustSDN may be useful for selecting the right trust policy in an SDN when considering possible alternatives.

3.2. The Simulated Evolutions with the Trust Strategies. This section shows examples of evolutions of the simulations in each combination of strategy and input configuration, in order to understand some features of these simulations.

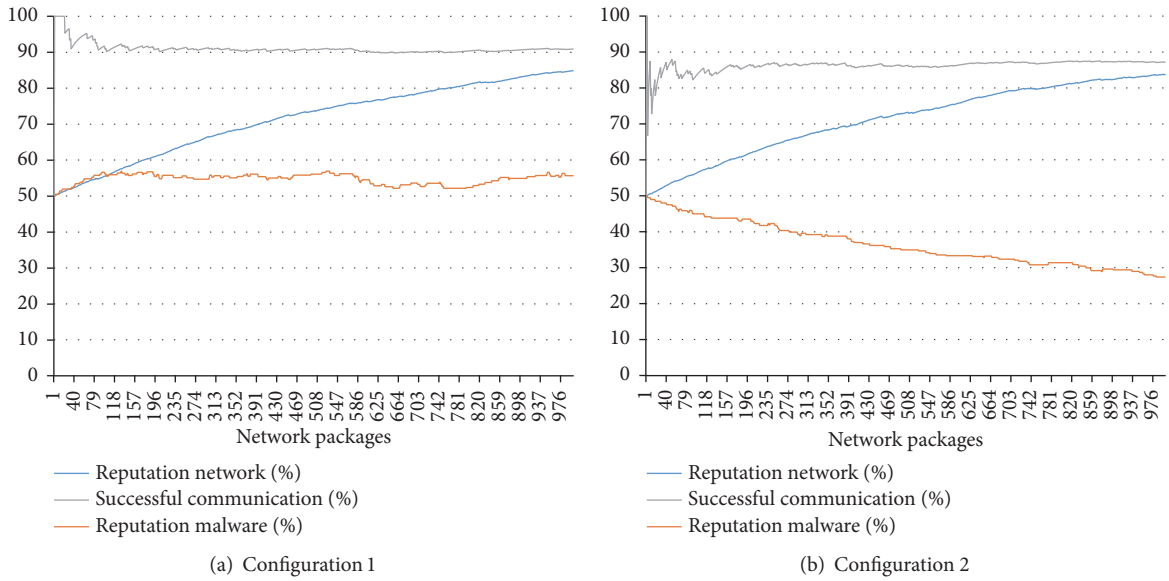
Figure 4 shows the evolution of the simulations of the Fixed strategy. One of the most relevant features is that it separates the reputation of network agents and malware agents very soon in the simulations (i.e., from the beginning). The drawback is that it starts decreasing the reputation of the normal network agents in the long term. The reason is that uncommon errors of these agents are taken seriously by this strategy, and wrongly isolating them enlarges the group of unfairly isolated network agents.

In Figure 5, one can observe that the most relevant aspect of simulations with the Tabsaond strategy is that the evolutions of strategies are smooth in comparison to the other strategies. In addition, Tabsaond strategy may reach stable states in different levels of reputations of malware regarding



(a) Configuration 1 (b) Configuration 2

Figure 4: Evolution of SDN metrics with Fixed strategy.



(a) Configuration 1 (b) Configuration 2

Figure 5: Evolution of SDN metrics with Tabsaond strategy.

the reliability of these agents, as suggested in the original TABSAOND approach.

Figure 6 shows the evolution of History strategy. One can observe that, in the beginning, it trusts almost the same malware agents and other agents, due to the minimum amount of interactions for analyzing an agent. Then, it almost separates perfectly the reputation of these two kinds of agents, getting really close to the ideal 0% and 100% trust for, respectively, malware agents and other agents in the second input configuration.

Therefore, ABS-TrustSDN presents detailed results of the simulations including their evolutions that have been useful for providing explanations to the outcomes of three different trust strategies. In general, the results of this experimentation advocate that ABS-TrustSDN may be an appropriate framework for designing and comparing trust strategies in SDNs, providing an adequate testbed simulation environment for this comparison.

4. Conclusions

The current work has introduced a novel agent-based framework for defining strategies for conforming trust models about network components. This framework and the corresponding ABS tool allow SDN designers to simulate and estimate the repercussion of certain protocols regarding the

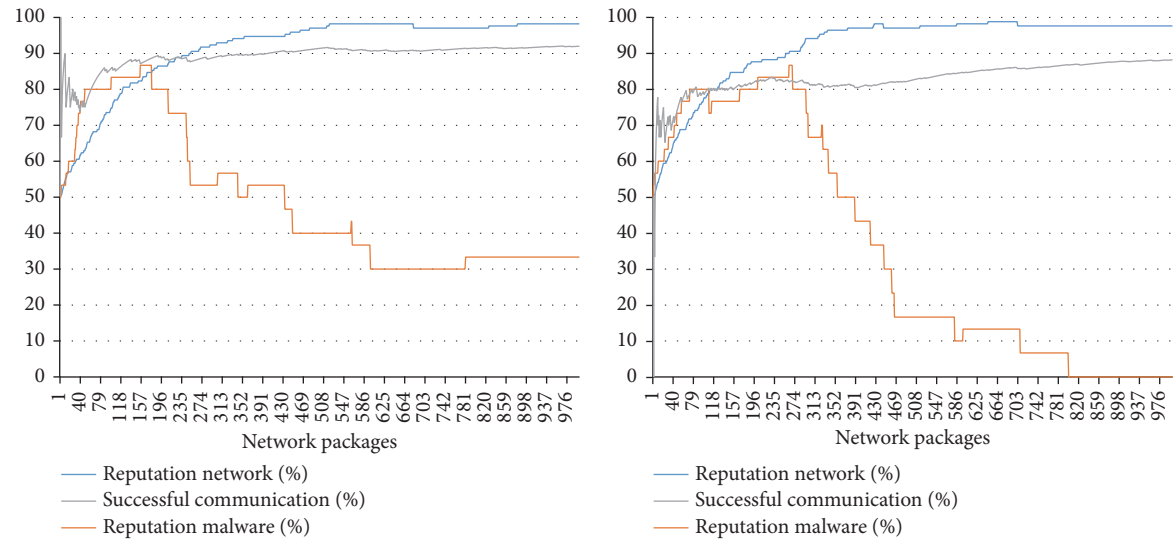


Figure 6: Evolution of SDN metrics with History strategy.

trust policy of the centralized controller of the SDN. This way, one can test several trust models in different contexts like short-term or long-term repercussions and the frequency of misbehavior of the network components with malware. This way, the current ABS supports the decision in selecting the most appropriate trust policy.

The current approach has assisted the definition of three different trust models. Each of these has been tested with two contexts with different frequencies of misbehavior of network components. Each case was simulated 1,000 times, and the results were compared. The results showed significant differences with a significance level of .001. In most cases, Cohen's d effect sizes were large according to Cohen's guidelines. Thus, the presented ABS approach was useful to find significant differences in the final simulated outcomes between different trust models for isolating problematic resources. The presented novel ABS-TrustSDN tool outputs several charts in its UI for analyzing the results not only at its final state but also in the evolution of the simulation. The evolution was crucial for understanding facts such as why the best isolation mechanism according to the final trust model does not necessarily imply the highest rate of success in the communications of the whole analyzed period.

The current work is planned to be enhanced by defining a higher number of trust policies in SDNs and simulating their repercussions. The presented approach will also be further experienced by simulating the repercussions of the variations of certain internal parameters of the strategies on the final simulated outcomes. The current approach will include the principles of model-driven development with (1) an agent-based metamodel [26] for supporting trust modeling and (2) model transformations for supporting the systems generation from their models [27]. The presented tool is planned to be tested with SDN designers as users to detect design opportunities related to aspects such as (a) improving the usability of the tool, (b) including more functionalities, (c) adding new ways of presenting and analyzing the data, and (d) supporting more primitives for supporting the definition of trust strategies.

Conflicts of Interest

The authors do not have any conflicts of interest regarding this work.

Acknowledgments

The authors would like to acknowledge "Desarrollo Colaborativo de Soluciones AAL" project (Ref. TIN2014-57028-R) supported by the Spanish Ministry of Economy and Competitiveness. This work was also supported by "Fondo Social Europeo" and the "Departamento de Tecnología y Universidad del Gobierno de Aragón" (Ref-T81).

References

[1] J. R. Ballard, I. Rae, and A. Akella, "Extensible and scalable network monitoring using OpenSAFE," in *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, INM/WREN'10*, p. 8, April 2010.

[2] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *Proceedings of the Local Computer Networks (LCN), 2010 IEEE 35th Conference*, pp. 408–415, IEEE, October 2010.

[3] T. Xing, Z. Xiong, D. Huang, and D. Medhi, "SDNIPS: enabling software-defined networking based intrusion prevention system in clouds," in *Proceedings of the 10th International Conference on Network and Service Management, CNSM 2014*, pp. 308–311, IEEE, Rio de Janeiro, Brazil, November 2014.

[4] S. Shin and G. Gu, "CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How

to provide security monitoring as a service in clouds?)," in *Proceedings of the 2012 20th IEEE International Conference on Network Protocols, ICNP 2012*, IEEE, Austin, TX, USA, November 2012.

[5] J. Naous, R. Stutsman, D. Mazieres, N. McKeown, and N. Zeldovich, "Delegating network security with more information," in *Proceedings of the In Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, pp. 19–26, 2009.

[6] A. K. Nayak, A. Reimers, N. Feamster, and R. Clark, "Resonance: dynamic access control for enterprise networks," in *Proceedings of the In Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, pp. 11–18, August 2009.

[7] A. Goodney, S. Narayan, V. Bhandwalkar, and Y. H. Cho, "Pattern based packet filtering using NetFPGA in DETER infrastructure," in *Proceedings of the 1st Asia NetFPGA Developers Workshop*, Daejeon, Korea.

[8] Z. Yan, P. Zhang, and A. V. Vasilakos, "A security and trust framework for virtualized networks and software-defined networking," *Security and Communication Networks*, 2015.

[9] P. Michiardi and R. Molva, "Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," in *Advanced Communications and Multimedia Security*, pp. 107–121, Springer, New York, NY, USA, 2002.

[10] J. Liu and V. Issarny, "Enhanced reputation mechanism for mobile ad hoc networks," in *Proceedings of the In International Conference on Trust Management*, pp. 48–62, Springer, Berlin, Germany, 2004.

[11] K. K. Fullam, T. B. Klos, G. Muller et al., "A specification of the agent reputation and trust (art) testbed: experimentation and competition for trust in agent societies," in *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 512–518, July 2005.

[12] D. Rosaci, G. M. L. Sarné, and S. Garruzzo, "Integrating trust measures in multiagent systems," *International Journal of Intelligent Systems*, vol. 27, no. 1, pp. 1–15, 2012.

[13] D. Jelenc, R. Hermoso, J. Sabater-Mir, and D. Trček, "Decision making matters: A better way to evaluate trust models," *Knowledge-Based Systems*, vol. 52, pp. 147–164, 2013.

[14] H. Chen, H. Wu, X. Zhou, and C. Gao, "Agent-based trust model in wireless sensor networks," in *Proceedings of the 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD '07)*, vol. 3, pp. 119–124, IEEE, Qingdao, China, July 2007.

[15] W. S. Kim, "Effects of a trust mechanism on complex adaptive supply networks: an agent-based social simulation study," *Journal of Guangxi Traditional Chinese Medical University*, vol. 12, no. 3, pp. 56–58, 2009.

[16] I. García-Magariño and R. Lacuesta, ABS-TrustSDN website. Available at http://webdiis.unizar.es/~ivangmg/abstrustsdn/ (last accessed July 28, 2017).

[17] 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.

[18] I. García-Magariño, M. Cossentino, and V. Seidita, "A metrics suite for evaluating agent-oriented architectures," in *Proceedings of the 25th Annual ACM Symposium on Applied Computing, SAC 2010*, pp. 912–919, Sierre, Switzerland, March 2010.

[19] R. Fuentes-Fernández, I. García-Magariño, J. J. Gómez-Sanz, and J. Pavón, "Integration of web services in an agent-oriented methodology," in *Proceedings of the Integration of Web Services in an Agent-Oriented Methodology*, vol. 3, pp. 145–161, 2007.

[20] C. Gutierrez and I. Garcia-Magariño, "A metrics suite for the communication of multi-agent systems," *Journal of Physical Agents*, vol. 3, no. 2, pp. 7–14, 2009.

[21] B. Desmarchelier and E. S. Fang, "National Culture and Innovation diffusion. Exploratory insights from agent-based modeling," *Technological Forecasting and Social Change*, vol. 105, pp. 121–128, 2016.

[22] I. García-Magariño, J. J. Gómez-Sanz, and J. R. Pérez-Agüera, "A multi-agent based implementation of a Delphi process," in *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2008*, vol. 3, pp. 1543–1546, International Foundation for Autonomous Agents and Multiagent Systems, Estoril, Portugal, May 2008.

[23] 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.

[24] B. L. Welch, "On the comparison of several mean values: an alternative approach," *Biometrika*, vol. 38, no. 3/4, pp. 330–336, 1951.

[25] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, Lawrence Earlbaum Associates, Hillsdale, NJ, USA, 2nd edition, 1988.

[26] I. García-Magariño, R. Fuentes-Fernández, and J. J. Gómez-Sanz, "A framework for the definition of metamodels for computer-aided software engineering tools," *Information and Software Technology*, vol. 52, no. 4, pp. 422–435, 2010.

[27] I. García-Magariño, S. Rougemaille, R. Fuentes-Fernández, F. Migeon, M. P. Gleizes, and J. Gómez-Sanz, "A tool for generating model transformations by-example in multi-agent systems," in *Proceedings of the In 7th International Conference on Practical Applications of Agents and Multi-Agent Systems PAAMS 2009*, vol. 55 of *Advances in Intelligent and Soft Computing*, pp. 70–79, Springer, Berlin, Germany, 2009.