Hindawi
Security and Communication Networks
Volume 2018, Article ID 4123736, 10 pages
https://doi.org/10.1155/2018/4123736

Research Article

Security Analysis of Dynamic SDN Architectures Based on Game Theory

Chao Qi, Jiangxing Wu, Guozhen Cheng, Jianjian Ai, and Shuo Zhao

National Digital Switching System Engineering & Technological R&D Center, Zhengzhou, Henan 450002, China

Correspondence should be addressed to Chao Qi; 13937147170@163.com

Received 26 September 2017; Accepted 26 December 2017; Published 23 January 2018

Academic Editor: Zhiping Cai

Security evaluation of SDN architectures is of critical importance to develop robust systems and address attacks. Focused on a novel-proposed dynamic SDN framework, a game-theoretic model is presented to analyze its security performance. This model can represent several kinds of players' information, simulate approximate attack scenarios, and quantitatively estimate systems' reliability. And we explore several typical game instances defined by system's capability, players' objects, and strategies. Experimental results illustrate that the system's detection capability is not a decisive element to security enhancement as introduction of dynamism and redundancy into SDN can significantly improve security gain and compensate for its detection weakness. Moreover, we observe a range of common strategic actions across environmental conditions. And analysis reveals diverse defense mechanisms adopted in dynamic systems have different effect on security improvement. Besides, the existence of equilibrium in particular situations further proves the novel structure's feasibility, flexibility, and its persistent ability against long-term attacks.

1. Introduction

SDN is a novel and promising framework which can be applied in traditional and wireless networks to achieve highly programmable switch infrastructure [1, 2]. It separates the data and control planes, which makes switches become simple data forwarding devices, and network is manipulated through logically centralized controllers [3]. It is no doubt that this processing mechanism will definitely improve the efficiency of network management and performance of network operation [4]. However, this reliance on a centralized controller can easily lead to a single point of failure if not carefully designed and implemented. Moreover, a new set of threats that are unique to SDN can render the network vulnerable especially when the control plane is compromised. Thus, in order to enhance security of SDN, different SDN architectures employing multiple controllers have been proposed, such as distributed controllers [5–10]. To the best of our knowledge, the most powerful dynamic security architecture for SDN is the Mcad-SA proposed in [9] which exploits heterogeneity, redundancy, and dynamism from multiple controllers to intensify security. However, no researches about their security performance evaluation have been conducted.

Our objective is to develop a general model to evaluate effectiveness of dynamic SDN architectures (take Mcad-SA as the instance) and provide some insights for designers to devise new, simple, and effective frameworks. To improve applicability, the model must have the ability to represent features of dynamic frameworks. Besides, it can capture the essential dynamic of progressive attack interacting with defense strategies to hinder that progression [11].

In this paper, we examine an abstract SDN-defense scenario designed to simulate strategic interactions between attacks and defense methods in dynamic architectures, as the control layer is a critical part in SDN and responsible for handling and distributing flows of information between network applications and the data plane. Thus we take the control plane's security as measurement of the whole SDN. Though simple, our model is generic, flexible, and applicable for a range of dynamic architectures and defense techniques. Our approach is game-theoretic which allows us to realize how effective are dynamic architectures when they deal with attacks and how security performance varies

as attackers and defenders change their behaviors. Unlike other game-theoretic researches, our model can apply to dynamic systems. It can also imitate systems which adopt specific defense means, like moving-target defense. And via systematic simulation, this empirical approach provides us with the opportunity to quantitatively estimate security performance of dynamic systems and defense technologies.

Among our findings, we characterize the control plane's security as the contention of controllers in it by opponents. And controllers' compromising process is proceeding procedurally, which means the probability of controllers being compromised is a function of the number of probes implemented on them. Besides, defenders have the capability to perceive the secure states of controllers since some dynamic architectures own detection mechanisms. Through experimental simulations, we observe that Mcad-SA are powerful against persistent probes from attackers. And reasonable defense and scheduling mechanism can further improve SDN's security. Besides, it is fascinating that system's detection capability is not as important as it is imagined in Mcad-SA since dynamism and redundancy can mitigate this weakness to a large degree.

The paper is organized as follows. The next section describes related work. Section 3 presents a detailed specification of our games. In Section 4, we present our game-theoretic experimental results. The last section concludes by summarizing our work and discussing future work.

2. Related Work

Recently, researchers have paid attention to security in SDN. On one hand, it aims at designing more resilient and robust controllers, such as FortNOX [12] and SE-Floodlight [13], are designed to deal with flow-rule conflict [14]. On the other hand, many SDN architectures with distributed controllers have been proposed to intensify SDN security from the point of framework. However, little work has been done on evaluating their security performance, especially when the architecture adopts moving-target defense to enhance security.

However, there is an amount of literature on computer security. In [15], formal methods have been used to provide an attack surface metric as an indicator of the system's security. So reducing attack surface is an effective approach to mitigate risk. Furthermore, [16, 17] pointed out launching a sequence of attacks is a common way for attackers to exploit vulnerabilities at multiple stages of the system. Thus, dividing the system into several layers and using attack graphs is a feasible method to assess the cause-consequence relationships between diverse network states. And it is a popular trend that proactive defense techniques are employed by the system to generate security strategies that alter over time to reduce the exposure of vulnerabilities and increase complexity and attack costs [18, 19]. Against such defense mechanisms, game theory provides an appropriate theoretical framework for modeling the win-lose situation between a defender and attacker [20]. In [21], FlipIt game, where two players compete for control of single resource, is designed to describe uncertainty in state of control. Extensions of FlipIt have considered additional actual scenario features. In an extension called "FlipThem" [22], multiple servers are incorporated and authors consider some extreme situations where attackers have to compromise one or all servers to achieve goals. Reference [11] refines the scenario further. It devises a more flexible utility model to allow payoffs between objectives of control and availability. Meantime it considers imperfect probe detection which indicates the defender observes attack probe actions with probabilities. Moreover it provides a much richer set of attack and defense strategies to evaluate overall system state. Although above work is related to security analysis, it cannot be applied in SDN environment directly, because SDN is a novel framework and has its own traits. For instance, its primary goal is to guarantee validity of flow rules delivered to switches. Besides, its defense strategy is backup or switch instead of reimage. Thus, we further make improvements on existing models and apply them into the SDN scenarios.

3. Game Specification

In our game the attacker and defender compete for the control of M controllers. Because once controllers are compromised then the whole network is also manipulated by attackers. M is changeable since the number of running controllers is varying with time in Mcad-SA. At first, controllers are in control of defenders. The attacker attempts to wrest control of a controller through via inserting some malicious flow rules, which needs probe actions on controllers. The success probability is relevant to the number of probes. Meantime, the defender may take some defense strategies based on current SDN frameworks. For instance, it may switch to another backup controller after it discovers the controller runs abnormally. Based on above analysis, first we introduce the abstracts of Mcad-SA and controller and then present models of players in detail later.

3.1. Abstract of Mcad-SA. In this section, we attempt to represent the control plane in Mcad-SA with a model. The reason why we take Mcad-SA as the example is that it is a typical representation of dynamic SDN architecture which employs heterogeneity, redundancy, and dynamism to improve security and owns perception simultaneously. The overview of Mcad-SA is presented in Figure 1. It consists of a variant control plane, a sensor, a scheduler, and an arbitrator. The novel control plane is equipped with multiple controllers. And the scheduler will select several as running controllers via specific mechanisms. In the meantime, the sensor can perceive current state of each controller to some extent. For example, it has the ability to realize how many probes have been conducted on controllers by attackers. As to the arbitrator, it determines the valid and correct flow rules down to switches.

The basic elements of the control plane in Mcad-SA include the number of controllers, the running controllers set each time, whether it has perfect perception, and what

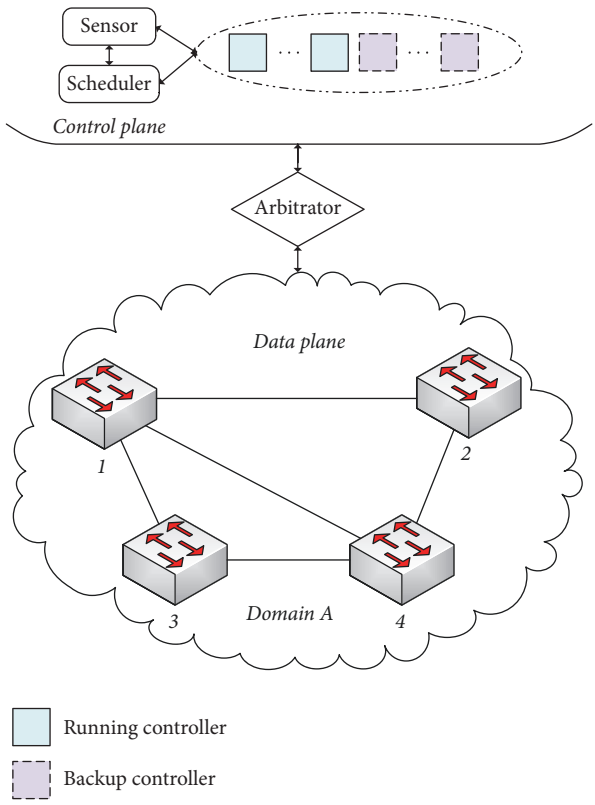

Figure 1: An overview of Mcad-SA.

scheduling strategies it adopts. Formally, abstract model of the control plane is a quadruple $\langle c, r, \psi, \Omega \rangle$, where

(i) $c \in [1\ N]$ represents how many controllers the control plane has and N is a very big positive number;

(ii) $r \in [1\ c]$ represents number of running controllers; r is changeable and generally an odd number;

(iii) $\psi \in \{0, 1\}$ indicates whether the control plane can sense the states of all controllers ideally; $\psi = 1$ indicates it can detect every probe the attackers implement, while $\psi = 0$ means it detects probe with some probability, and the probability is related to number of probes already on controllers;

(iv) Ω illustrates sets of strategies the scheduler employs, such as random selection or other scheduling algorithms.

Generally, there are $2|\Omega|$ occasions according to above settings. For instance, when $\psi = 1$ and Ω is random selection, which indicates the system can be aware of every probe precisely and it chooses controllers randomly as the next running controller set. Next, we introduce the model of players.

3.2. Model of Player. In order to quantitatively measure security performance of the control plane, we need define information of players. Here we focus on their actions, utilities, and strategies.

3.2.1. Actions. Before describing players' actions, we depict the state of a controller at first, which is convenient for modeling players. At any point of time, controllers' state can be represented by who takes control of it, what type, whether it is up or down, and how much progress the attacker has made towards controlling if the attacker has begun his probe action. Formally, controller state is a triple $\langle \chi, \kappa, s, \rho \rangle$, where

(i) $\chi \in \{\text{att}, \text{def}\}$ represents the player who controls the controller;

(ii) κ represents what kind of controller it belongs to, such as floodlight [23], Ryu [24], or OpenDaylight [25].

(iii) $s \in \{\text{up}\} \cup \{\text{down}\}$ represents whether the controller is running ($s = \text{up}$) or is not up ($s = \text{down}$);

(iv) ρ is the successful number of probes attackers has carried out on a controller.

The state of the overall SDN system is determined by the joint state of all running controllers, plus the current time t.

For attackers, its action is called *probe*. To describe the actions precisely, let $\langle \chi_t, \kappa_t, s_t, \rho_t \rangle$ be the state at time t. And we denote $\langle \chi_{t+}, \kappa_{t+}, s_{t+}, \rho_{t+} \rangle$ as the state after the actions. Thus the probe action's effect can be specified via the following rules:

(i) If $s_t = \text{down}$, the probe action has no effect: $\langle \chi_{t+}, \kappa_{t+}, s_{t+}, \rho_{t+} \rangle = \langle \chi_t, \kappa_t, s_t, \rho_t \rangle$.

(ii) If $s_t = \text{up}$ and $\kappa_{t+} = \kappa_t$, the number of probes is incremented: $\rho_{t+} = \rho_t + 1$, while $s_t = \text{up}$ and $\kappa_{t+} \neq \kappa_t$, which indicates, during the probe, the running controller has been replaced with another type of controller. In Mcad-SA, this switching mechanism can intensify the security of control plane to some degree since they have different bugs, which requires more cost to attack successfully. So the number of probes is incremented with probability $1-e^{-\beta\rho_{t+}}$ under this occasion because formal effective probe may be invalid on another type of controller. And when $\chi_t = \text{att}$, then the attacker maintains control: $\chi_{t+} = \text{att}$. While $\chi_t = \text{def}$, the defender continues to control the controller with probability $e^{-\gamma\rho_{t+}}$. For attackers, its probability of control raises up to $1 - e^{-\gamma\rho_{t+}}$. This probability can be seen as the probability with which the controller generates malicious flow rules, where $\beta > 0$ and $\gamma > 0$ are scaling parameters.

The defender has two actions. One is reimage and the other is switch. The goal of reimaging controllers is reset its state to initial settings. So the defender wins control back and the cumulative effect of probe is eliminated. And the state is reset as follows: $\langle \chi_{t+}, \kappa_{t+}, s_{t+}, \rho_{t+} \rangle = \langle \text{def}, \kappa_t, \text{down}, 0 \rangle$. The switch action is to replace the running controller with another controller, which can reduce the probability of control by attackers to some extent. And the state is transited to that of new controller: $\langle \chi_{t+}, \kappa_{t+}, s_{t+}, \rho_{t+} \rangle = \langle \chi_t', \kappa_t', s_t', \rho_t' \rangle$.

3.2.2. Utility. In SDN, the major function of controllers is producing valid flow rules and delivering them to switches.

And the purpose of the defender is to guarantee the control plane generates as many right flow rules as possible, while the attacker' objective is just the opposite. Thus we measure each player's payoff based on the state of flow rules the control plane creates.

We presume the number of effective flow rules a healthy controller produces is n_e^h. If s_t = down, the controller cannot generate any valid rules. While if s_t = up, the number of valid rules n_e^p is related to probes the attacker conducts (i.e., ρ_t). It can be calculated via

$$n_e^p = n_e^h e^{-\gamma\rho_t}. \tag{1}$$

Then we can define each player's payoff. Given the running controller set C_R, the total payoff of the attacker is

$$\mathrm{AP}_{\mathrm{total}} = \sum_{c_i \in C_R} n_e^h \left(1 - e^{-\gamma\rho_t}\right) - \Phi_{\mathrm{ac}}, \tag{2}$$

where c_i is the ith controller and Φ_{ac} is the total attacking cost and proportional to the number of probes. Correspondingly, the defender's payoff can be computed via

$$\mathrm{DP}_{\mathrm{total}} = \sum_{c_i \in C_R} n_e^p - \Phi_{\mathrm{dc}}, \tag{3}$$

where Φ_{dc} is the total defending cost. It includes the cost of reimaging and switching to other controllers and is proportional to the number of reimages and switches.

3.2.3. Strategies. A strategy for players is to choose controllers on which they execute their actions. They are always triggered by observed events or detected information.

The attacker's strategy consists of two factors: how many probes they can execute each time and the number of probes they have conducted on each controller. Here we consider a strategy defined by the following policy rules:

(i) *Random-Probe (RanProb):* select uniformly among running controllers under defender's control (χ_t = def).

(ii) *Maximum-Probe (MaxProb):* select controllers that have been probed most to proceed next probe.

For the defender, the condition of executing reimage is similar to attackers. Also it has the following strategies:

(i) *Random-Reimage (RanReimage):* pick up running controllers randomly to reimage regardless of their states.

(ii) *Maximum-Reimage (MaxReimage):* choose controllers that have been probed most to be reimaged.

As to the other action of defender, the switch action is executed under two mechanisms. One is after a fixed time interval Δt, regardless of whatever states all running controllers are. The other is according to the fraction of controllers regulated by defenders. When it falls below a threshold τ, the switching process is evoked. However, when $\psi = 0$ the defender cannot realize controllers' state directly. Instead, the sensor component can assist the control plane estimating their conditions. Let n_c^{def} denote the estimated number of controllers in the hand of defenders; if

$$\frac{\mathbb{E}\left[n_c^{\mathrm{def}}\right]}{|C_R|} < \tau, \tag{4}$$

then the switch action is activated. One detail to be mentioned, the switch strategy may have more than one method according to system settings. Here, we make specific explanations on switch strategies (Ω). We list three common kinds of scheduling methods below. And we let C_n be the next set of running controllers.

(i) *RandomWithRepeat:* select C_n uniformly from all the available controllers.

(ii) *RandomWithoutRepeat:* select C_n uniformly from the rest controllers to guarantee $C_n \cap C_R = \emptyset$.

(iii) *MaxSG*: this is a perceptive scheduling strategy proposed in [26] to maximize security gain of the control plane during each switch. For a controller, it will be switched preferentially to another controller which is belonged to another type, deployed on another host and probe least. That is to say, this mechanism ensures two controllers have maximum difference and the next running controller is the most reliable one.

4. Game-Theoretic Analysis

In this part, we conduct simulations on Mcad-SA based on above assumptions. First, we present a brief introduction of its operating mechanisms again.

In Mcad-SA, the running controller set is varying with time under this situation. Moreover, the system can adjust its switching strategy based on specific settings. That illustrates that dynamism is introduced into SDN.

4.1. Simulation Environments. It is obvious that this game is a periodic process. So we let this game last for T time units ($T = 200$). For the control plane employing more than one controller, we take $M = 7$. The scaling parameters β and γ equal 0.1. The fixed switch interval Δt is set as 25 and the reimage interval is 20. And we let τ equal 0.5.

Next, we implement experiments using a discrete-event simulator. Before the simulation, we define f_p as failure probability of the control plane. f_p is related to conditions of all running controllers. Only when over $\lceil(|C_R| + 1)/2\rceil$ controllers are compromised simultaneously will the system fail. For controller $c_i \in C_R$, its reliability r_i^c equals the ratio of valid flow rules (n_e^p) to total rules it produces (n_e^h). Then f_p can be computed via (5), where $C_{c \geq \lceil(|C_R|+1)/2\rceil}$ represents the set of all situations with more than $\lceil(|C_R| + 1)/2\rceil$ controllers being compromised simultaneously. C_c^u is the set of benign controllers in C_c. The smaller f_p is, the more right flow rules the control plane can produce, which means more powerful ability to resist attacks. Here, we let n_e^h be 1000:

$$f_p = 1 - \sum_{C_c \in C_{c \geq \lceil(|C_R|+1)/2\rceil}} \prod_{c_i \in C_c^u} \left(1 - r_i^c\right) \prod_{c_j \in C_c \setminus C_c^u} r_j^c. \tag{5}$$

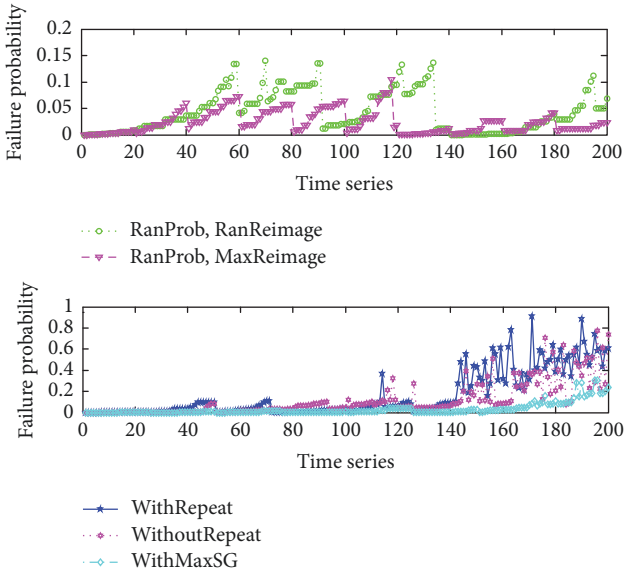

Figure 2: Failure probability under respective defending strategies when facing random probes.

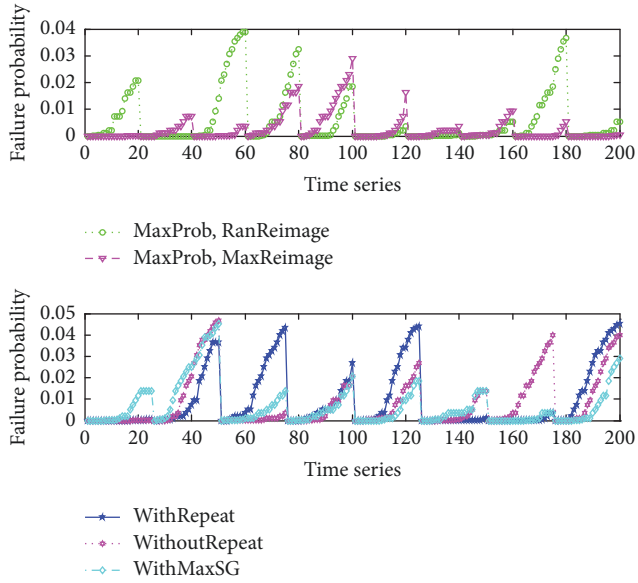

Figure 3: Failure probability under respective defending strategies when facing maximum probes.

4.2. Simulation Process. At first, we assume that all controllers are perfect without any probes. Thus, f_p is 0 at the beginning via (1). Then attackers will choose an attacking strategy to conduct attacks, which results in the rise of f_p. With f_p being higher and higher, the system is going to launch respective defense mechanisms to reduce f_p. As a result, f_p is varying with time and players' strategies. Through analyzing the changing trend of f_p which is related to status of running controllers, we can realize the real-time state of Mcad-SA under various combinations of players' tactics. Similarly, players' payoffs can be obtained according to probes on controllers and cost of players via (2) and (3). Then we can conclude that which player owns the control of the system.

Further, in order to measure how powerful Mcad-SA is, we classify the scenarios of Mcad-SA into two situations according to ψ. When $\psi = 1$, we call it perfect probe detection environment, while $\psi = 0$, it is regarded as imperfect probe detection environment. Then we analyze the simulation results under two circumstances when adopting various scheduling strategies (Ω).

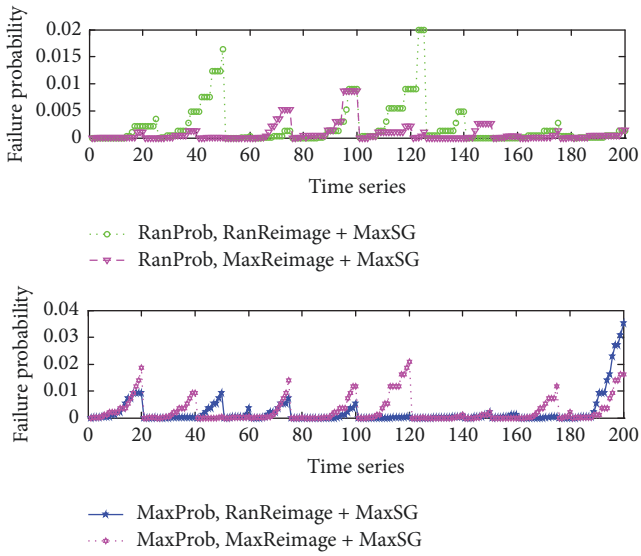

Figure 4: Failure probability with associate defending strategies under perfect detections.

4.3. Simulation Results

4.3.1. Perfect Probe Detection Environment. In perfect probe detection environment, the defender is able to observe all probes from attackers on controllers.

(a) Respective Defending. First, we make comparisons on the effectiveness of respective defending strategies when facing different attacking means. Figure 2 indicates variations of system's security performance when attackers adopt random probe, while Figure 3 illustrates the situations when attackers employ maximum probes.

In Figure 2, the upper picture is the result when taking reimage actions only while the map below is the consequence when taking switch actions only. It is obvious that when the defender adopts reimage methods to defend random probes, random and maximum reimages have similar effect (MaxReimage is slightly better). However, the consequences of various switching strategies are very different. When the system employs random switching (with repeat or without repeat), security performance is close. While adopting *MaxSG*, reliability of the system is strengthened to some degree, especially after 100 time units. The reason is that probes of controllers will increase to a large number with time going on. So under this situation, perceptive switch can ensure the system select more reliable controllers (probed least) than random switch. This phenomenon further demonstrates superior scheduling ways have better effect on improving system's security performance.

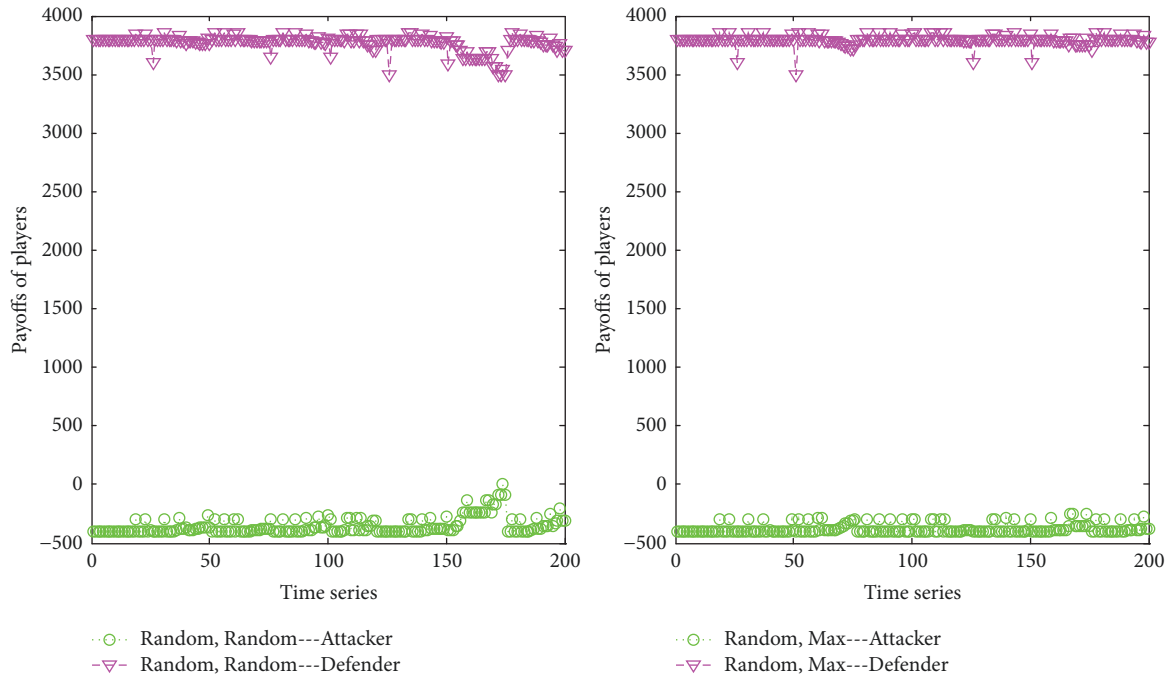

Figure 5: Payoffs of players against random probes with associate defending strategies under perfect detections.

In Figure 3, it is another situation where the attacker conducts maximum probes. An interesting phenomenon is observed that all defending measures have approximate preventive effect. That is due to the cause that this attack mode results in severe destruction on specific controllers. Thus once defending methods can avoid selecting these controllers, the security state of the system can be guaranteed. So it comes to a conclusion that unless attackers have to probe particular controllers to acquire important information, it is of little value to carry out persistent maximum probes.

(b) Associate Defending. As is analyzed above, *MaxSG* is the most effective switch option compared to random switch. Then we combined it with reimage action to testify associate defending performance. That is to say, the system adopts reimage and switch behaviors simultaneously.

Figure 4 illustrates the results of associate defending when encountering random and maximum probe individually. The upper section is the consequence against random probe. It is apparent that associate defending mechanism significantly reduces the failure probability of the system and enhances its security gain as the value is decreased with an order of magnitude (from 0.05 to 0.005 averagely), while the bottom part is the case against maximum probe. Similarly, security of the system is intensified to some extent and the value of failure probability is lowered to half of that in previous circumstance.

(c) Players' Payoffs. Next, we analyze players' payoffs in occasions where associate defending is used to deal with probes from attackers. Figure 5 is the case with random probe while Figure 6 is the situation with maximum probe.

In Figure 5, the left part shows how payoffs vary with random reimage and *MaxSG* while the right section is the result with maximum reimage. Both pictures manifest that payoffs of players are steady, which means attackers cannot acquire much valuable information from the system and defenders can maintain it operating in a relatively secure state meantime. As to some downward pulse, their appearances are due to the trigger of reimage or switch actions. However, strictly speaking, defense effect of the right one is superior to that of the left one since the line of the attacker is more stable and the value is lower, which further proves maximum reimage is better than random reimage under this environment.

Figure 6 is similar to the shape of Figure 5 except that downward pulses are more obvious. That is an indication that the system is severely compromised since several controllers have been probed persistently under maximum probes. But in other occasions, the payoff of the attacker is less than that in Figure 5, which again demonstrates maximum probe is not a good choice unless attackers have special purposes.

Actually, the described process can be regarded as an approximate zero-sum game. Once the attacker gains some privilege, the defender loses some control and vice versa. Based on above simulations and analysis, the following conclusion can be drawn: introducing dynamism to SDN is a feasible and effective way to mitigate attacks' impact and improve the system's robustness. And whatever actions attackers take, the best response for defenders is maximum reimage and *MaxSG*. In other words, eliminating and restoring the most damaged components in the system is a critical point to ensure its security.

4.3.2. Imperfect Probe Detection Environment. Then we analyze the results when the system does not own the capability to detect each probe ($\psi = 0$, i.e., imperfect detection).

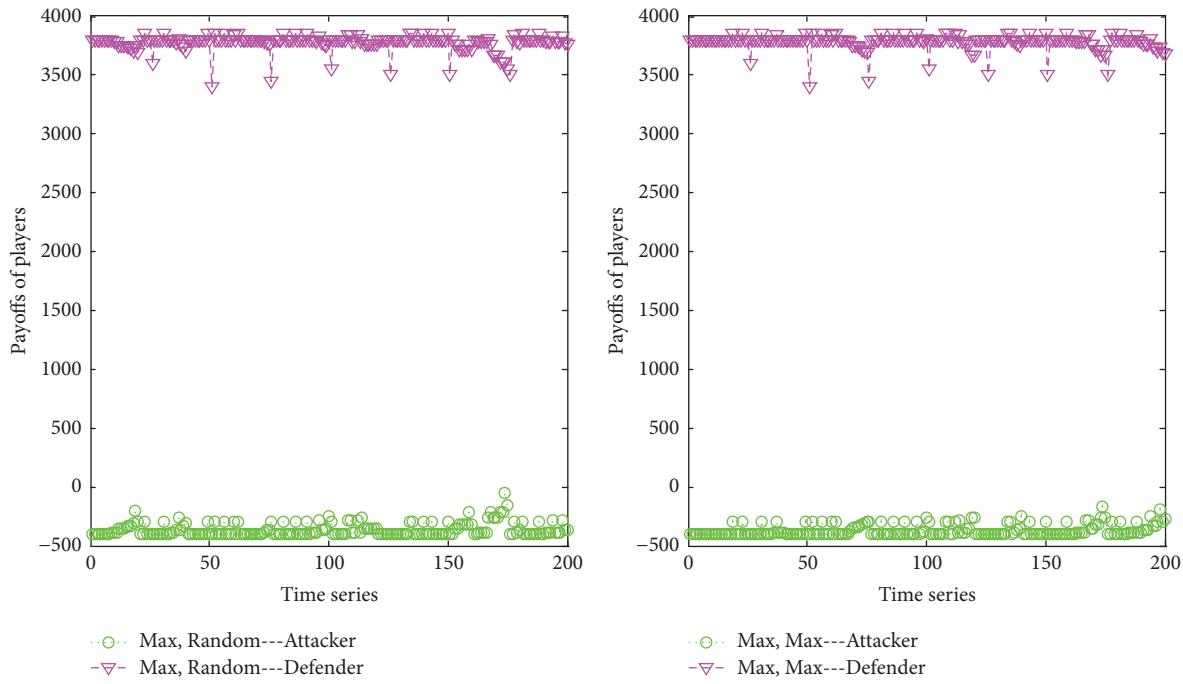

Figure 6: Payoffs of players against maximum probes with associate defending strategies under perfect detections.

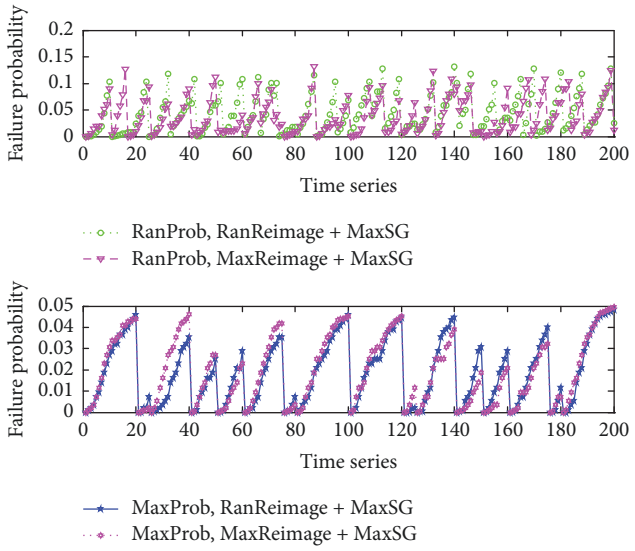

Figure 7: Failure probability with associate defending strategies under imperfect detections.

Here, we neglect the analysis on respective defending since its changing trend is similar to that in perfect probe detection environment. So we pay attention to associate defending and players' payoffs.

(a) Associate Defending. Figure 7 is the simulation under imperfect probes. Compared to perfect probe, the most distinction is the failure probability increases overall. Actually, the system's security performance under two associate defenses is approximate. The cause resulting in this phenomenon is that the system can no longer perceive the real state of controllers, which may incur the scheduler to make an incorrect policy. For instance, the scheduler seems to choose a controller which has been probed least to be the next running controller, but actually this one has already been severely compromised. This phenomenon may also appear in the reimage situations. Above decisions will influence the system's reliability to some extent. However, a point to be noticed, although without perfect probe detection, the failure probability is also maintained at a low level because of reimage and switch strategies.

(b) Players' Payoffs. Figures 8 and 9 are the variations of players' payoffs under random and maximum probes, respectively. The distribution of payoff is fluctuating extensively, particularly in random probe. That is due to the reason that defenders are likely to discover the controllers being continuously compromised when attackers employ maximum probes even under imperfect detections. Thus, we can notice downward pulses are more distinct in Figure 9 than Figure 8, which is strong evidence on the effectiveness of reimage and switch strategies in maximum probes. Meanwhile, the attacker's payoff is clearly increased in both figures. Nevertheless, it is still kept in a stable interval, which demonstrates players reach an equilibrium. Moreover, for attackers, the results further authenticate that adopting random probe is superior to maximum probe in general.

According to above analysis, it is evident that detection capability has effect on the security performance of the system to some extent since inaccurate state information of controllers will definitely influence the decision-making process of the scheduler. And ways of reimage and switch

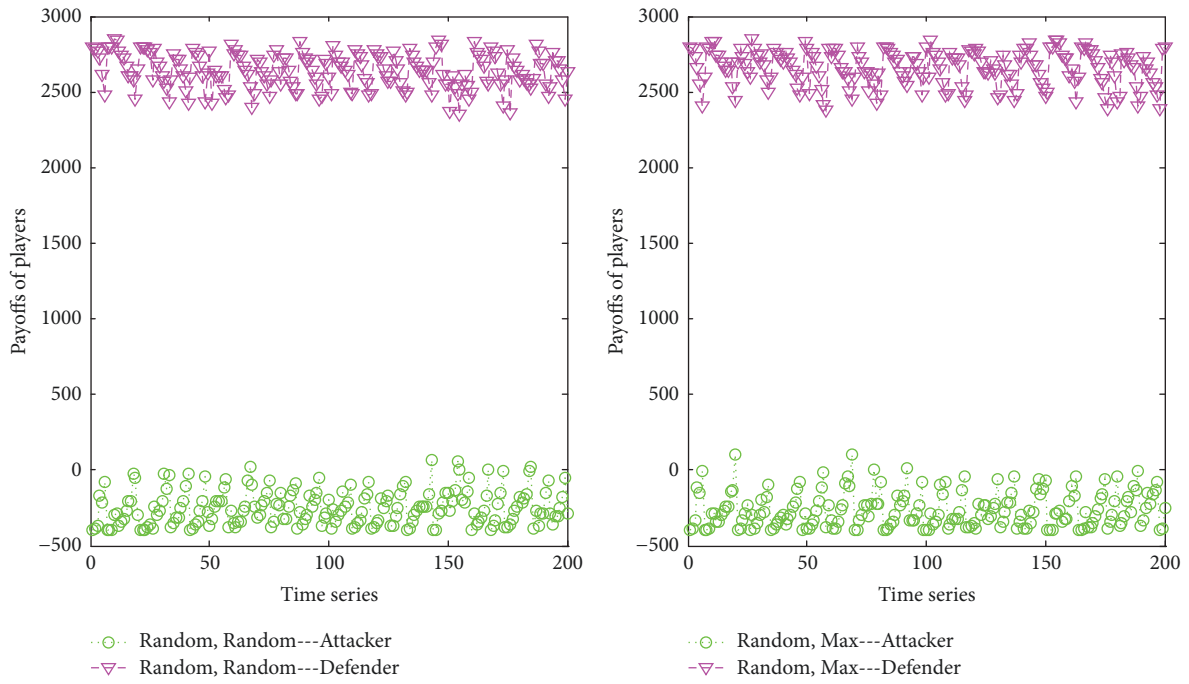

Figure 8: Payoffs of players against random probes with associate defending strategies under imperfect detections.

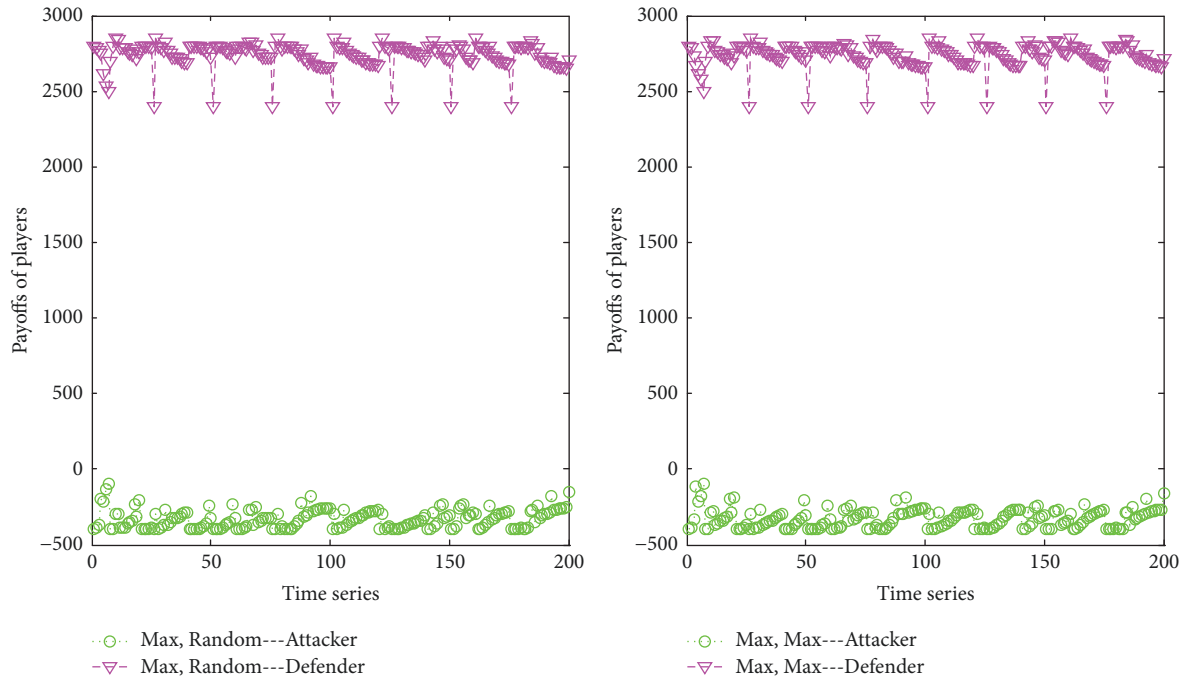

Figure 9: Payoffs of players against maximum probes with associate defending strategies under imperfect detections.

can lead to diverse defense results. In most cases, random probe is a better option for attackers while the best response of defenders is the combination of maximum reimage and *MaxSG*. However, as long as dynamism, heterogeneity, and redundancy are introduced into SDN, its security can be improved significantly and the failure probability of the control plane can be maintained at relatively a low standard.

4.4. Equilibrium Results. In this section, we report the equilibria found in the simulation. As we all know, when the system reaches an equilibrium, the failure probability of Mcad-SA and players' payoffs stays stable. They have no intent to alter their strategies.

Actually in Mcad-SA, whenever in perfect or imperfect probe detection environment, the situations of equilibria are

identical. The only occasion in which the equilibrium exists is that defenders adopt maximum reimage and MaxSG scheduling method while attackers select maximum probe. From the simulation results above, as long as attackers carry out maximum probe, they always can acquire higher payoffs and higher failure probability of the system over random probe on matter what actions defenders take. Similarly, defenders will tend to choose maximum reimage and MaxSG to obtain higher payoffs and lower failure probability whenever they face random or maximum probe. In other words, maximum reimage and MaxSG are the best choice for defenders and maximum probe is the best option for attackers as they can achieve their goals to maximum extent.

5. Conclusion

Evaluation of security performance of SDN architectures plays a critical role in designing reliable structures and estimating system risks. Focused on Mcad-SA, a novel SDN structure, we attempt to establish a model to analyze its ability to resist attacks with the assistance of game theory. This model can represent players' related information (actions, strategies, etc.) and quantitatively assess system's capability to deal with risks. Experimental results indicate the introduction of dynamism, redundancy, and heterogeneity into SDN can intensify system's security and maintain its consistent capability against diverse probes which is an extraordinary weakness in current SDN frameworks. Further, we analyze equilibrium in particular situations where common types of probes and defense methods are included. And analysis results present hints that for dynamic systems the design of their defense strategies has certain effect on security enhancement, which implies that how to devise effective defense mechanism is crucial to maximize security gain. In the future, we plan to take internal security mechanism of diverse controllers into consideration and pay attention to hybrid security analysis.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (no. 61521003), the National Key R&D Program of China (no. 2016YFB0800100 and no. 2016YFB0800101), and the National Natural Science Foundation of China (no. 61602509).

References

[1] N. McKeown, T. Anderson, H. Balakrishnan et al., "OpenFlow: enabling innovation in campus networks," *Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[2] D. He, S. Chan, and M. Guizani, "Securing software defined wireless networks," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 20–25, 2016.

[3] C. Monsanto, J. Reich, and N. Foster, "Composing software-defined networks," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, pp. 1–14, USENIX Association, 2013.

[4] Z. Guo, M. Su, Y. Xu et al., "Improving the performance of load balancing in software-defined networks through load variance-based synchronization," *Computer Networks*, vol. 68, pp. 95–109, 2014.

[5] P. Berde, M. Gerola, J. Hart et al., "ONOS: towards an open, distributed SDN OS," in *Proceedings of the 3rd ACM SIGCOMM 2014 Workshop on Hot Topics in Software Defined Networking (HotSDN '14)*, pp. 1–6, August 2014.

[6] V. Yazici, M. O. Sunay, and A. O. Ercan, "Controlling a software defined network via distributed controllers," in *Proceedings of the 2012 NEM Summit*, pp. 16–20, 2014.

[7] H. Li, P. Li, S. Guo, and A. Nayak, "Byzantine-resilient secure software-defined networks with multiple controllers in cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 4, pp. 436–447, 2014.

[8] X. Jin, J. Gossels, J. Rexford, and D. Walker, "CoVisor: a compositional hypervisor for software-defined networks," in *Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI '15)*, pp. 87–101, USENIX Association, May 2015.

[9] C. Qi, J. Wu, H. Hu et al., "An intensive security architecture with multi-controller for SDN," in *Proceedings of the 35th IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS '16)*, pp. 401-402, April 2016.

[10] K. ElDefrawy and T. Kaczmarek, "Byzantine fault tolerant software-defined networking (SDN) controllers," in *Proceedings of the 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC '16)*, pp. 208–213, June 2016.

[11] A. Prakash and M. P. Wellman, "Empirical game-theoretic analysis for moving target defense," in *Proceedings of the 2nd ACM Workshop on Moving Target Defense (MTD '15)*, pp. 57–65, 2015.

[12] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for OpenFlow networks," in *Proceedings of the 1st ACM International Workshop on Hot Topics in Software Defined Networks (HotSDN '12)*, pp. 121–126, August 2012.

[13] P. Porras, S. Cheung, M. Fong, K. Skinner, and V. Yegneswaran, "Securing the Software Defined Network Control Layer," in *Proceedings of the Network and Distributed System Security Symposium (NDSS '15)*, San Diego, CA, USA, 2015.

[14] Z. Guo, Y. Xu, M. Cello et al., "JumpFlow: Reducing flow table usage in software-defined networks," *Computer Networks*, vol. 92, pp. 300–315, 2015.

[15] P. K. Manadhata and J. M. Wing, "An attack surface metric," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 371–386, 2011.

[16] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using Bayesian attack graphs," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 1, pp. 61–74, 2012.

[17] C.-C. Ten, C.-C. Liu, and M. Govindarasu, "Vulnerability assessment of cybersecurity for SCADA systems using attack trees," in *Proceedings of the IEEE Power Engineering Society General Meeting (PES '07)*, pp. 1–8, Tampa, FL, USA, June 2007.

[18] S. Jajodia, A. K. Ghosh, and V. Swarup, *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*, Springer Ebooks, 2011.

[19] S. Jajodia, S. K. Ghosh, V. S. Subrahmanian, V. Swarup, C. Wang, and X. S. Wang, *Moving Target Defense II: Application of Game Theory and Adversarial Modeling, Advances in Information Security*, Springer, 2012.

[20] M. H. Manshaei, Q. Zhu, T. Alpcan, T. Basar, and J.-P. Hubaux, "Game theory meets network security and privacy," *ACM Computing Surveys*, vol. 45, no. 3, article 25, 2013.

[21] M. Van Dijk, A. Juels, A. Oprea, and R. L. Rivest, "FlipIt: The game of "stealthy takeover"," *Journal of Cryptology*, vol. 26, no. 4, pp. 655–713, 2013.

[22] A. Laszka, G. Horvath, M. Felegyhazi, and L. Buttyán, "FlipThem: modeling targeted attacks with flipit for multiple resources," in *Decision and Game Theory for Security*, vol. 8840 of *Lecture Notes in Computer Science*, pp. 175–194, Springer International Publishing, Cham, 2014.

[23] FloodLight, "Open SDN controller," http://floodlight.openflowhub.org.

[24] "Ryu SDN Framework," http://osrg.github.io/ryu.

[25] "OpenDaylight Consortium," http://www.opendaylight.org.

[26] C. Qi, J. Wu, H. Hu, and G. Cheng, "Dynamic-scheduling mechanism of controllers based on security policy in software-defined network," *IEEE Electronics Letters*, vol. 52, no. 23, pp. 1918–1920, 2016.