Hindawi
Security and Communication Networks
Volume 2018, Article ID 6751042, 13 pages
https://doi.org/10.1155/2018/6751042

Research Article

Duo: Software Defined Intrusion Tolerant System Using Dual Cluster

Yongjae Lee, Seunghyeon Lee, Hyunmin Seo, Changhoon Yoon, Seungwon Shin, and Hyunsoo Yoon

KAIST, 291 Daehak-ro, Yuseong-gu, Daejeon 34141, Republic of Korea

Correspondence should be addressed to Seungwon Shin; claude@kaist.ac.kr

Received 30 September 2017; Accepted 18 December 2017; Published 21 January 2018

Academic Editor: Qiang Fu

An intrusion tolerant system (ITS) is a network security system that is composed of redundant virtual servers that are online only in a short time window, called exposure time. The servers are periodically recovered to their clean state, and any infected servers are refreshed again, so attackers have insufficient time to succeed in breaking into the servers. However, there is a conflicting interest in determining exposure time, short for security and long for performance. In other words, the short exposure time can increase security but requires more servers to run in order to process requests in a timely manner. In this paper, we propose Duo, an ITS incorporated in SDN, which can reduce exposure time without consuming computing resources. In Duo, there are two types of servers: some servers with long exposure time (White server) and others with short exposure time (Gray server). Then, Duo classifies traffic into benign and suspicious with the help of SDN/NFV technology that also allows dynamically forwarding the classified traffic to White and Gray servers, respectively, based on the classification result. By reducing exposure time of a set of servers, Duo can decrease exposure time on average. We have implemented the prototype of Duo and evaluated its performance in a realistic environment.

1. Introduction

Nowadays, (nearly) everything in our lives is digitalized and connected to each other, and it makes us easily access necessary information. However, the richness of the connectivity causes another side effect that motivates cybercriminals to reach connected resources for malicious purposes. To minimize this effect, researchers and practitioners have been putting huge amount of effort into devising diverse network security approaches. Those approaches are commonly classified into two types: (i) signature-based network attack detection system (e.g., Snort [1]) and (ii) network anomaly detection system [2]. No one has doubt that they have effectively and efficiently protected our network environments so far.

However, cyberattacks have become more sophisticated and existing network intrusion detection/prevention mechanisms are no longer effective against such advanced attacks [3]. In response to this problem, researchers have proposed network intrusion tolerant system (ITS), which proactively defends victim systems without detecting nor blocking intrusion attempts [4, 5]. Of the various types of ITS proposed until today, recovery-based ITS, which periodically reverts the production services to their clean initial states, is now widely endorsed and deployed [3, 6, 7].

Technically, the recovery-based ITS can effectively reduce attack surfaces of the servers [8] by making the servers available to the public including any potential adversaries for a very short period of time and restoring possibly compromised servers. Such a time period is referred to as *exposure time*, and it plays an extremely crucial role in recovery-based ITS [7, 9, 10]

The exposure time in recovery-based ITS creates a trade-off between the cost and security. Taking the short exposure time leads a server to be more intrusion tolerant, but it will require more backup servers for high service availability [7]. In contrast, the longer exposure time will require less computing resources, accordingly. If the exposure time is

configured to be too long, it will increase the chances of the server being compromised. To this end, finding the optimal exposure time for a recovery-based ITS is an important problem.

Despite its importance, the matter of finding the optimal exposure time is understudied and still remains unanswered. The previous studies have only focused on reducing attack surfaces of target servers [8, 11] or efficiently recovering target servers [12, 13]. Indeed, they have pointed out that finding the best exposure time for ITS is a critical problem that must be solved, but this problem has never been clearly addressed.

In this paper, we propose Duo, a recovery-based ITS that leverages software defined networking (SDN) and network function virtualization (NFV) technologies to minimize the overall exposure time without consuming additional computing resources. Unlike the previous ITS proposals that applied uniform exposure time for all the servers, Duo adaptively controls the exposure time. To do this, Duo first classifies network requests into two criteria, *benign* and *suspicious*, using the existing network security systems (e.g., NIDS). The servers on the network are also classified into two server groups, *White and Gray*, and the servers that belong to the White group only handle the benign requests, while the servers in the Gray group handle the rest. Here, the Gray group in Duo is assigned shorter exposure time than that of the White group because it is more likely that the Gray servers will be compromised, and thus the servers are more frequently reverted to their initial state for security purposes.

Although Duo can reduce the systemwide exposure time, it introduces new challenges that must be solved. One challenge is that the traffic classification points become the network bottlenecks. Since our system investigates all the incoming network packets, the network will likely be unstable or even unavailable whenever there is a burst of inbound network traffic. Another challenge is that it should be able to determine how many and which type of servers (Gray or White) should be spawned or killed based on the volumes of suspicious and benign traffic on the network.

In this work, we address the first challenge with the help of SDN [14, 15] that is inherently designed to control network flows flexibly. Duo forwards traffic to a corresponding server according to its type. In SDN, this flow control is easily described in a flow rule; once the rule is activated, successive network flow will be forwarded to appropriate servers. Furthermore, in resolving the bottleneck problem of traffic classification, we employ NFV technology that allows deploying multiple traffic classifiers as a virtual appliance in a distributed manner. In addition, to resolve the second challenge, we propose a novel optimized resource management algorithm that is based on integer linear programming (ILP). When compared to the heuristic server provisioning scheme proposed in [12], our algorithm finds the optimal number and combination of different types of servers within the budget instead of relying on the optimistic assumption that limitless servers are available.

The main contributions of the paper are as follows: (1) we propose a new recovery-based ITS architecture that can reduce the average exposure time by classifying network flows into two types and treating them differently according to their type. (2) To handle each type of network flows differently and to minimize the overhead incurred by the classification process, we introduce SDN/NFV into ITS. To the best of our knowledge, Duo is the first attempt to incorporate ITS into SDN/NFV technologies. (3) We implement a prototype system to evaluate the performance of Duo, and the evaluation result shows that Duo effectively reduces the average exposure time.

The rest of this paper is organized as follows: Section 2 presents preliminaries of ITS and SDN, and related work is given in Section 3. Section 4 describes the architecture of Duo and explains how it works in detail. The performance of Duo is evaluated in Section 5, and we discuss its limitation in Section 6. Finally, Section 7 presents future work and concludes this paper.

2. Background

2.1. Intrusion Tolerant System. Intrusion tolerant system (ITS) is a new type of security framework that ensures the service availability and the integrity of potential victim systems, which could be affected by any type of intrusion attempts [9, 10, 21]. Unlike conventional security systems such as Intrusion Detection and Prevention Systems (IDPS) whose protection effectiveness solely depends on their intrusion detection capability, ITS takes a different approach that does not rely on the intrusion detection techniques in protecting potential victim systems.

The key idea of ITS is that it continuously restores the potential victim systems to their original pristine condition. By doing so, on the one hand, the victim systems can keep providing services even if they have been compromised, and on the other hand, it can fundamentally minimize any potential collateral damage that may be caused by the compromise. For example, production servers usually stay online for several months or even years without shutting down [7], and thus, in a common APT (Advanced Persistent Threat) attack scenario, they are the most preferred targets to compromise and take control over because the attackers can keep a long-term access to them and perform insider attacks against other assets within the security perimeter. ITS can fundamentally block such an illegitimate long-term access to the internal systems that could put the entire network and asset at risk by continuously reverting the production servers.

Meanwhile, maintaining high service availability is as important as ensuring the system integrity; however, ITS's continuous system restoration strategy significantly impacts the service availability. To solve this problem, ITS maintains a number of server clones for each service, and it uses one copy for providing the actual service at a time while having the rest of the copies as the backups in the original pristine state [7]. When the deployed instance expires and goes through the restoration process, replacing the expired instance, one of the backups is immediately deployed, thus enabling a transparent and smooth service handoff. This feature can be easily implemented using virtualization technology. For instance, virtual machines (VMs) could be leveraged to maintain system clones and handoff the service from one clone to another. In addition, compared with restoring physical

Table 1: Summary table of related work.

Related work	Topic	Description
SCIT [7, 16]	ITS	Initially proposed recovery-based ITS
ACT [12, 13]	ITS	Recovery-based ITS with adaptive cluster scaling
CloudWatcher [17]	SDN/NFV	Network flow inspection framework using SDN technology
QoSE [18]	SDN/NFV	On-demand security service provisioning in an SDN/NFV environment
Bohatei [19]	SDN/NFV	SDN/NFV-based DDoS defense system
Snort [1] Suricata [7]	IDS	Open source network intrusion detection system
HIF [20]	NTC	History-based IP filtering for SIP protection

machines, restoring virtual machines are far more time- and cost-efficient.

2.2. Software Defined Networking. Software defined networking (SDN) is a new networking technology that enables centralized network management. In SDN, the control plane is decoupled from the network devices and placed on a centralized controller, which can maintain a global network view. The controller is often implemented in software, and it is responsible for making decisions on how to deal with network flows, while the data plane simply forwards the packet based on the decisions made by the controller. SDN controllers also implement a network abstraction interface, which allows implementing diverse and innovative network functions into SDN applications. For this reason, SDN networks are known to be dynamic, flexible, and programmable.

Network managers can administer the network system flexibly and dynamically. For example, the managers are able to define a new flow rule that forwards network flow incoming from switch A to switch B. If the SDN controller installs the flow rule on the data plane, then the data plane delivers network packets as instructed in the rule. In the traditional network architectures, this process of rule generation and registration is not possible without additional efforts.

Recently, the characteristics of SDN are investigated to find a new possibility that can enhance security performance. In particular, combined with network function virtualization (NFV) [22], SDN attracts much of the attention of security researchers and network device vendors because the combination achieves successes in defending against network attacks including a DoS (denial of service) attack defense and network anomaly detection [17–19, 23].

2.3. Incorporating ITS into SDN. Since Duo serves network traffic separately, it is a natural choice to incorporate the system into SDN. In other words, to solve the bottleneck problem that can be caused by the traffic classification, we distribute the traffic classifiers within the system network. It would be difficult to control incoming traffic to go through the distributed traffic classifiers in the legacy network environment. In SDN, however, the SDN control plane can generate flow rules to forward incoming traffic to one of the deployed traffic classifiers, which will be selected in the way of minimizing routing overhead.

3. Related Work

We now discuss previous studies that have addressed the challenging issues similar to ours and present summaries of them in Table 1.

3.1. Intrusion Tolerant System. The Self-Cleansing Intrusion Tolerance (SCIT) architecture restores its VM servers in the system to their known pristine or initial state in a periodic fashion [7, 16]. A server stays active for a certain period of time, called exposure window, and the SCIT controller recovers the server to its initial state after the exposure window expires. As a consequence, SCIT makes it impossible for an intrusion to reside in the system longer than the predefined exposure window, and thereby damage caused by the intrusion can be minimized.

An intrusion tolerant system based on adaptive cluster transformation (ACT) attempts to guarantee a high level of system availability by transforming its VM cluster in an adaptive manner [11, 12]. A dynamic cluster expansion/reduction scheme, the main feature of this approach, examines whether the volume of incoming requests increases or decreases. If the volume increases due to an explosion of normal traffic or a DDoS attack, it decides to expand its VM cluster by adding more VMs. On the other hand, if the request volume decreases, it reduces the VM cluster by subtracting VMs from the cluster to save its computing resources and prepare for additional attacks.

These ITS architectures seek to gain safety and availability, but they are not yet comprehensive enough to satisfy both security attributes simultaneously. In other words, SCIT does not account for attacks to exhaust computing resources such as network bandwidth or CPU, which are the major threat to modern network systems. ACT-based ITS, on the other hand, focuses on availability rather than safety, and it is improper for large-scale server systems because a threshold should be calculated in advance to change the cluster size, and it adds or drops only 2 VMs at a time.

3.2. Software Defined Networking. Combination of SDN and NFV technologies contributes to the enhancement of network security systems. CloudWatcher monitors network flows and selects optimal routing paths for network flows to be inspected by security devices [17]. To this end, CloudWatcher's routing selection algorithm is implemented on

top of SDN that has well-defined functionalities to control network flows flexibly.

QoSE is a network security framework that provides security services in an adaptive fashion employing NFV [18]. Specifically, QoSE framework is composed of multiple virtualized network function middleboxes, and considering not only path status but also the middleboxes' availability, QoSE forms paths through which network flows should pass.

Bohatei is a DDoS defense system that is also based on SDN and NFV [19]. Unlike conventional DDoS defense mechanisms, Bohatei virtualized DDoS defense appliances based on NFV and distributed them flexibly. Hence, network flows do not have to go through a certain security middlebox that is fixed at a certain point, which can act as a bottleneck point.

Inspired by CloudWatcher and QoSE, Duo adopts distributed traffic classifiers so that we address the latency problem that can be caused by the centralized traffic classification process. In addition, Bohatei motivates us to design the NFV-based traffic classifier, which helps efficiently distribute the classifiers over the data plane.

3.3. Intrusion Detection System. Intrusion detection system (IDS), such as Snort [1] and Suricata [24], is a reactive security system that defends a target system against intrusions (or successful attacks) by detecting attack trials [2]. It detects an attack trial and reports that incident to the network administrator, and then he or she decides which action to take to eliminate the threat. IDS can employ different approaches that are classified as anomaly- and signature-based detection in general [25], and both of them require signature or anomalous behavior patterns of attacks in advance to detect attacks. However, it is not often the case that such signatures and anomalous characteristics of attacks are available before the attacks show up, and thus IDS sometimes fails in detecting attack trials.

Different from IDS, the recovery-based ITS takes a proactive approach to eliminate threats from attackers. Aiming to remove any intrusions in the target system by self-rejuvenating methods [16], the recovery-based ITS assures the safety of the systems.

3.4. Network Traffic Classification. Network traffic classification has been an important research topic in network security community [26]. In our work, network traffic classification is deemed a process to seek a client's reputation in terms of security. In this context, our traffic classification is analogous to Peng and others' work [20] in that their history-based IP filtering admits incoming packets if their senders have already accessed a network's edge router normally and their IP addresses have been registered in a history-based IP database. However, if an adversarial user launches attacks after deceiving the history-based IP database by sending reconnaissance packets, then it is hard for history-based IP filtering to block attack packets. Our work, on the other hand, takes a conservative approach that inspects every incoming packet to reinforce the security performance.

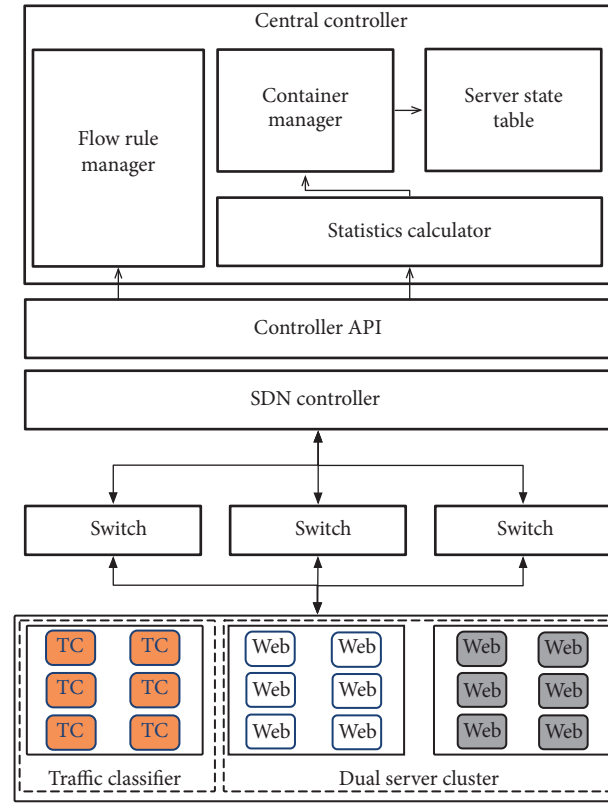

Figure 1: The architecture of Duo.

4. System Design

Duo is a new network intrusion tolerant system (ITS) that consolidates the security features of ITS with SDN. Since the main goal of Duo is to minimize exposure time of ITS by treating network requests differently, we employ SDN to control network requests flexibly in realizing Duo.

4.1. Architecture Overview. In Figure 1, we illustrate the architecture of Duo which is composed of three main components as follows: (i) central controller, (ii) dual server cluster, and (iii) traffic classifier.

Central controller manages the dual server cluster and SDN flow rules. To this end, the central controller has four modules as follows:

(i) *Statistics calculator* collects statistics on how many network flows in each type (i.e., suspicious and benign) are getting in to the system.

(ii) *Container manager* computes the optimal size of the dual server cluster and the proper number of traffic classifiers, using the statistics information provided by the statistics calculator. This module also creates container servers and traffic classifiers.

(iii) *Server state table* keeps track of servers' state change that follows the server lifecycle model.

(iv) *Flow rule manager* creates and installs a new flow rule if unknown network flow accesses the system.

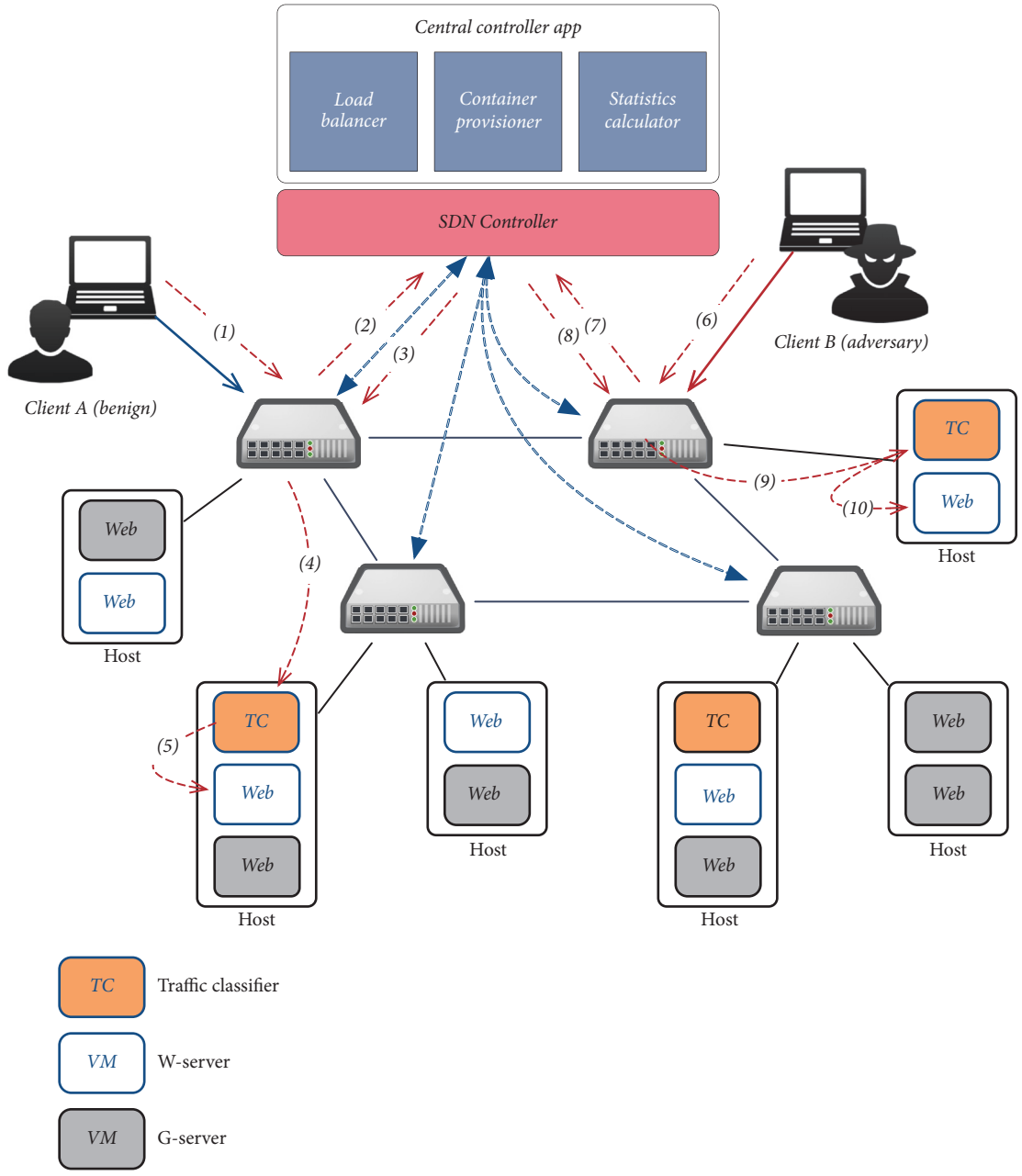

Figure 2: The overall working scenario before an attack is detected.

Moreover, this module updates already installed flow rules if benign network flows turn out to be malicious.

Dual server cluster is two groups of the servers, White and Gray clusters, and the servers in both clusters have the same functionalities but the exposure time. The two clusters are adaptively scaled as the central controller computes the appropriate size of each cluster in a periodic fashion.

Traffic classifier (TC) is a software appliance that determines whether network traffic is benign or not. On top of the NFV technology, Snort [1], an open source NIDS, is installed on multiple containers, which are distributed over the data plane. So, whenever a network flow enters the system, it should be inspected by one of the traffic classifiers.

4.2. Overall Working Scenarios. Now, we present the overall workflow of Duo by taking example scenarios as shown in Figure 2. When (1) an HTTP request arrives at Duo, the data plane searches for a flow rule that instructs how to handle the request. If the data plane cannot find any matching rules for the request, (2) it asks the control plane (or, more specifically, the central controller) what to do with the request, sending a `packet_in` message. Then, (3) the central controller installs a flow rule on the data plane, which sends the request to the TC

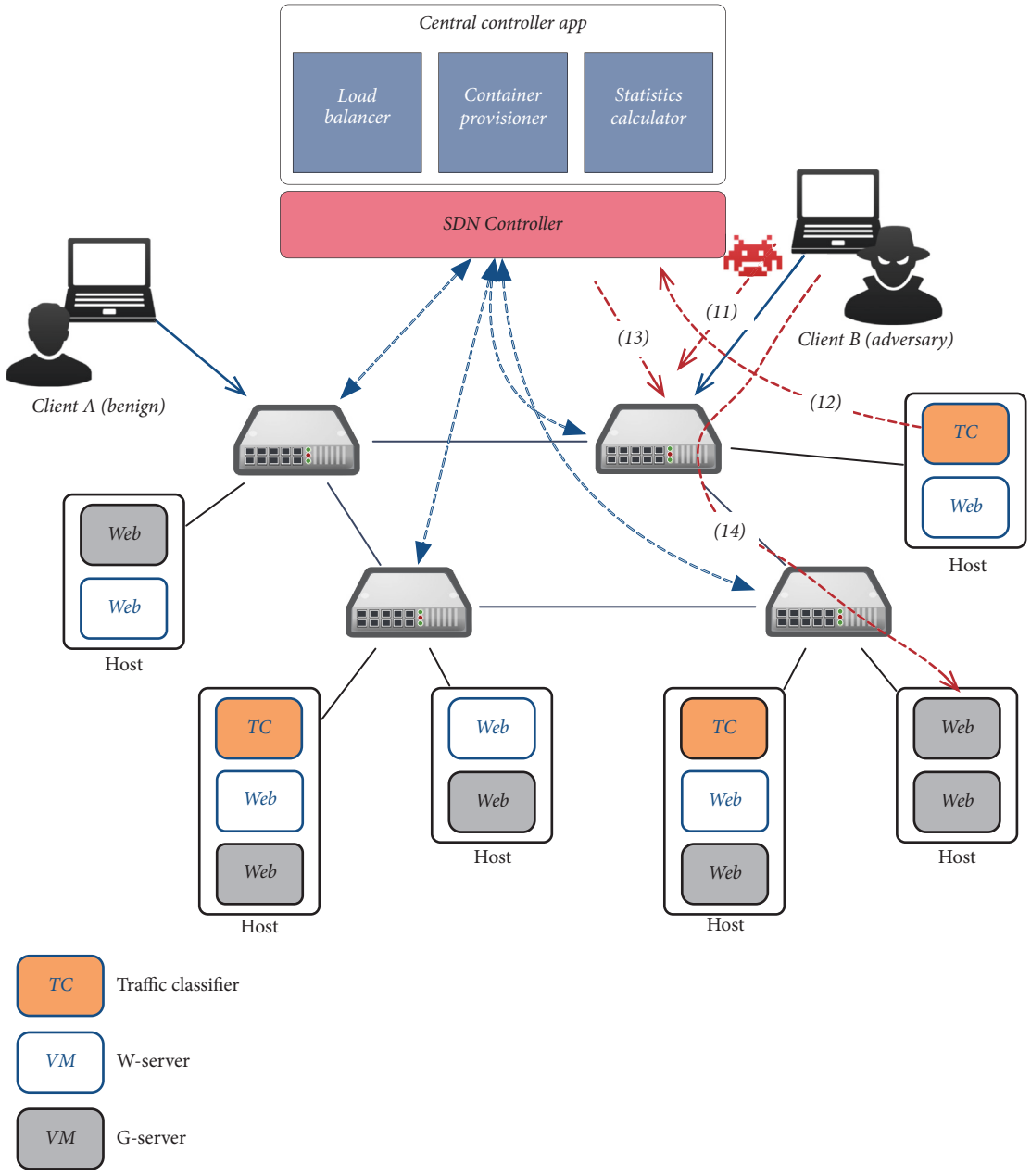

Figure 3: Working scenario when an attack trial is detected.

using a `flow_mod` message. (4) The TC inspects the request and decides the type of the request between two types: benign and suspicious. Suppose that (5) the request is determined as benign by the traffic classifier, and then this request is sent to a White server.

On the other hand, suppose that an attacker wants to access a server. In this case, Duo handles his packets as usual (i.e., from (6) to (10) in Figure 2). However, at some time point, as shown in Figure 3, (11) he will start an attack, and this trial will be recognized by the traffic classifier. Then, (12) the accident is reported to the central controller immediately, and (13) the previously installed flow rules for the attacker will be removed from the data plane. The central controller will install a new flow rule that describes (14) to forward the attacker's packets to a Gray server afterward.

4.3. Why ITS Classifies Network Flows. The key idea of Duo is to have different recovery schedules for the components (i.e., servers). Basically, Duo is a recovery-based ITS that recovers the servers to their pristine state in a periodic manner. However, unlike existing ITS studies, Duo does not employ a uniform recovery schedule. Rather, it classifies traffic into two types and gives the servers different exposure times depending on the type of traffic to which the servers are assigned.

Then, why do we bother to classify network flows even in designing the recovery-based ITS? The answer is that every flow does not leave the same footprint when it is served by the server. In other words, most of the users are benign and want to receive services reliably, but some adversarial users seek to make security breaches by infecting the server. Thus, if the server served illegitimate or suspicious flows, it is highly probable that the server would be polluted while serving them. On the other hand, if the flows are benign, possibly the server remains uncontaminated.

Under this premise, we divide the servers into two groups: White and Gray cluster. The two clusters have a different recovery schedule because they are in charge of serving the two different types of network flows: benign and suspicious flow. Benign flows are served by the servers in the White cluster; hence the servers do not have to recover frequently, which helps save resources. Suspicious flows, on the contrary, are served by the servers in the Gray cluster, which will recover more often than those in the White cluster. This is because the servers in the Gray cluster are believed to have pollutants during their operation.

4.4. How Duo Fortifies Security of ITS. The goal of Duo is to reduce the exposure time of a recovery-based ITS. Duo has a separate group of servers (i.e., Gray cluster) that are in charge of serving suspicious network traffic, whereas the rest of the servers (i.e., White servers) will handle benign network traffic. Since the Gray servers are more likely to be exposed to threats than the White servers, we give the Gray servers much shorter exposure time than the usual exposure time, which is assigned to the White servers. Intuitively, as we have assigned the shorter exposure time to a part of the servers, the average exposure time over the system becomes much shorter than when all the servers have the usual exposure time.

To achieve the goal, we addressed three main issues as follows:

(i) Duo employs the dual server cluster that constitutes two types of servers, White and Gray clusters, that are separated by their exposure time. The Gray servers are given shorter exposure time than that of the White servers because they will handle suspicious traffic.

(ii) Duo scales the dual server cluster adaptively, to cope with the change in volume of network traffic. The scaling process is operated by an ILP-based algorithm, which finds the optimal size of each server cluster within budget in terms of the container servers available.

(iii) To classify network traffic into suspicious and benign types, Duo employs traffic classifiers that are implemented on top of the NFV technology. To minimize overhead that can be caused by traffic classification, multiple traffic classifiers are distributed over the network.

4.4.1. Dual Server Cluster. The dual server cluster is two server clusters that consist of containerized servers. Duo distributes the container servers between the dual server cluster according to their exposure time. If a server is assigned to the White cluster, then its exposure time is set to E_{Long}. On the other hand, if the server is allocated to the Gray cluster, then the server's exposure time becomes E_{Short}. The difference between E_{Long} and E_{Short} is time duration for the server to be online; E_{Short} is defined much shorter than E_{Long} because the servers in Gray cluster serve suspicious requests that are more likely to be malicious.

By letting exposure time of a group of container servers much shorter than that of the other, we are able to reduce the overall exposure time of the entire system. Existing ITS architectures assign all the VM servers uniform exposure time without addressing how they select exposure time [8, 11, 12, 16, 27, 28]. However, we argue that the overall exposure time can be reduced without using additional resources if we assign each server unidentical exposure time in accordance with the type of clients that the server will handle.

To show how exposure time of Duo can be reduced, we define $\overline{E}$, *systemwide exposure time,* which represents the mean exposure time of the running servers of ITS. Unlike existing ITS architectures employing uniform exposure time, we employ two different exposure times, and thus it is not possible to perform a direct comparison of exposure time between Duo and other architectures. Hence, we compare the two systemwide exposure times, $\overline{E}_{\text{Duo}}$ for Duo and $\overline{E}_{\text{Uni}}$ for ITS with uniform exposure time:

$$\overline{E}_{\text{Duo}} = \frac{E_{\text{Long}} \cdot W + E_{\text{Short}} \cdot G}{N}, \tag{1}$$

$$\overline{E}_{\text{Uni}} = E_{\text{Uni}}, \tag{2}$$

$$E_{\text{Short}} < E_{\text{Long}} = E_{\text{Uni}}. \tag{3}$$

We define $\overline{E}_{\text{Duo}}$ as shown in (1), where W and G denote the number of each type of the running servers, respectively, and N is the sum of W and G (i.e., the total number of the running servers). Note that, in case of an ITS architecture with uniform exposure time, $\overline{E}_{\text{Uni}}$ is equal to its server's uniform exposure time, E_{Uni}, as shown in (2). Here, if we leave E_{Long} the same as E_{Uni} and assign E_{Short} to the Gray cluster, which is much shorter than E_{Uni}; then we conclude that it holds that $\overline{E}_{\text{Duo}} < \overline{E}_{\text{Uni}}$ as long as (3) is satisfied.

4.4.2. Resource Management. One pivotal role the central controller plays is to manage resources, or the servers. More specifically, the central controller recovers the servers following their lifecycle model and provisions the dual server cluster for additional servers when required.

Server Lifecycle Model. Before giving a detailed explanation of the lifecycle model, we need to discuss the server state table first. The central controller has the server state table, where state information of running servers is stored. A server's state is summarized in seven fields as described in Table 2, and the central controller keeps track of the server's state change from creation to destruction. The entries in the table are used for scheduling server recovery and making flow rules to refer to a location of servers.

Table 2: An entry of the server state table.

Server id	Type	No. of recv packets	No. of sent packets	Creation time	State	Location

When the central controller decides to create a server, it also generates a table entry for that server. Initially, except the number of the received and sent packets, all the fields are filled with values of a unique id, the type (i.e., White or Gray), the creation time, the state (i.e., initial creation), and the location (e.g., physical host id), respectively.

Now, we present the lifecycle model in detail as illustrated in Figure 4. When a server is created, its state becomes *creation*, and the timestamp for this event is recorded in the state table entry. With this timestamp, we will determine whether a server should be recovered or not. After created, the server soon receives the first packet, and its state transitions to *running*, which means it is ready to handle packets. This state transition event is also recorded in the state table entry.

When the server only has the grace time left within its operational time, then the server goes into *grace period* state. Here, the grace time denotes a time window during which the server does not accept new requests and processes the remaining requests in its queue. Giving the grace time to the server, we prevent users from experiencing a sudden interruption to the service provision. Finally, the server's state will be changed from *grace period* to *destruction* after the grace time expires, or all the remaining requests are served even if the grace time is not expired. At this state, the server will be stopped and destroyed.

In SCIT, the servers' lifecycle model is also proposed, but their lifecycle model differs from ours in that our containerized servers do not need to remain ready for a long time because they can be instantly created and become ready [7].

Server Provisioning. Another role the central controller plays is to increase or decrease the size of server clusters dynamically. If the volume of traffic increases, Duo creates and puts new servers into the clusters as far as computing resources allow. Although defending against a denial of service attack is beyond the scope of this work, this operation ensures availability to provide users with services in a timely way. On the other hand, to save resources, Duo decreases the size of server clusters when the central controller estimates that the number of running servers is too large for the volume of current traffic.

When a server is going to recover, another server should replace the server as long as the volume of traffic remains flat. Similarly, in case of an increase in the volume of traffic, additional servers should be supplied promptly to cope with the situation. In Duo, to satisfy this requirement and to achieve the main goal of reducing the systemwide exposure time, we take an optimization approach to the resource management problem. To be specific, we formulate the problem as an integer linear program (ILP) that is aimed at guaranteeing availability with the systemwide exposure time minimized.

In (4), we present the ILP formula for the resource management problem with the objective function to minimize the sum of the running servers' exposure time. The

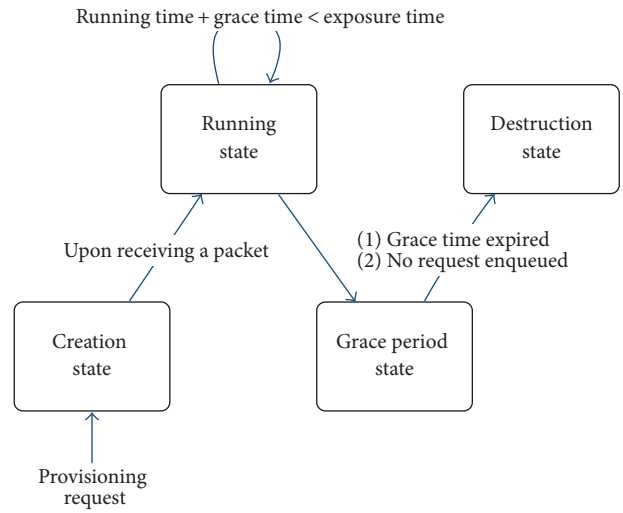

Figure 4: The lifecycle of a containerized server.

objective function is dependent on the variables x and y, which are the number of White and Gray servers in running state, respectively. K denotes a server's capacity to serve the requests, which means the maximum number of requests that a server can process for one second (rps). In addition, we define the volume of benign and suspicious traffic as BT and ST in rps, respectively:

$$\text{minimize} \quad \sum^{x} E_{\text{Long}} + \sum^{y} E_{\text{Short}} \tag{4}$$

subject to the following constraints:

$$x \geq \frac{\text{BT}}{K} \tag{5}$$

$$y \geq \frac{\text{ST}}{K} \tag{6}$$

$$x + y \leq \text{MAX} - T, \tag{7}$$

where

K is the capacity of a server (rps),

BT is the volume of benign traffic (rps),

ST is the volume of suspicious traffic (rps),

T is the number of traffic classifiers,

MAX is the max number of servers a system can make.

In order to handle requests without much delay, the system expects to have x White servers and y Gray servers. However, the expected total number of servers, $x + y$, should not exceed MAX $- T$, assuming that the system can have total

```
(1) def calcRank (switch, tc):
(2)    p = .5
(3)    state = tc.state
(4)    busy = tc.busyness
(5)    near = relativeDistance (switch, tc)
(6)    return state * (p * near + (1−p) * busy)
(7)
(8) def selectTC (switch):
(9)    ranks = list ()
(10)   for tc in trafficClassifiers:
(11)      rank = calcRank (switch, tc)
(12)      ranks.append (rank)
(13)
(14)   ranks = sorted (ranks, descendingOrder=True)
(15)   return ranks [0]
```

Algorithm 1: Traffic classifier selection algorithm.

MAX containers, out of which T containers are used as traffic classifiers. Satisfying these constraints, we periodically find the optimal values for the variables x and y.

4.4.3. Network Flow Management. Duo requires all new network flows to go through a TC before flow rules for the flows are activated. This approach, however, produces two challenging issues related to security and performance. First, an attacker can try to compromise the TC for the purpose of forwarding his malicious packets to the White cluster. Second, because the central controller checks the type of the flow and then installs a new rule for the flow, users may experience a long response delay if the network classification process takes much time to give a result.

We elaborate on two possible scenarios that an attacker prevents the TC from achieving its goal. First, we can imagine that the attacker compromises the TC in order to let the central controller make inappropriate flow rules. For example, if the attacker succeeds in taking control over the TC, he may make the classifier tag his packets as benign. Then, his packets will be forwarded to the White cluster, and he can get more time to attack the entire system afterward. The other scenario is that an attacker can launch resource exhaustion attacks (e.g., DoS attacks) to overload the TC with new network flows. In this case, since the TC is the entry point of the system, the throughput of the entire system can be dropped considerably.

Distributed Traffic Classifier. In order to address these issues, we (i) make the traffic classifiers intrusion tolerant and (ii) distribute them across the dual server cluster. Like the web servers, the TCs are also containerized and recovered to their clean state, which are basic intrusion tolerance defense operations that we believe can protect the TCs against attacks. Distributed TCs, on the other hand, are more related to availability. For instance, if the system has a single TC, then it cannot continue to provide users with services as soon as the TC fails. Therefore, we deploy multiple instances of the TC, as described in the previous section. However, we limit the number of TCs because they perform a single operation (i.e., classification).

To realize the distributed traffic classifiers, Duo deploys the TCs as a virtual network function (VNF) and distributes them across the dual server cluster. Recently, combined with SDN, the network function virtualization (NFV) technology fosters better security by enabling network functions to be virtualized. For example, without using a middlebox, we are able to install software IDS, such as Snort [1] or Suricata [24], on a VM and to initiate SDN flow rules in order for network traffic to make a detour through the VM. Then, network traffic will be inspected or monitored by the VM, more specifically the virtualized IDS.

Traffic Classifier Selection. If a packet arrives an edge switch of the system, and there does not exist a matching flow rule to the packet, then it is forwarded to a TC to be inspected. However, because there are distributed TCs, when we select a TC to classify the packet, we need to consider the routing overhead and the state of a TC. For example, Since the containerized TCs are distributed over the dual server cluster, there are many paths from the edge switch to a TC, and some of the paths may not be optimal. In addition, a TC can be busy when a large number of new packets are flocked to the classifier. Therefore, the central controller needs to make an efficient routing path for the new packets.

Therefore, we propose a TC selection algorithm as presented in Algorithm 1, considering the factors as follows:

(i) nearness: the relative distance from the edge switch to a TC,

(ii) busyness: the degree to which a TC is relatively busy,

(iii) state: the state of a TC, whether or not a TC is going to recover soon.

When we design the TC selection algorithm, we first consider how many nodes exist between the edge switch and a TC. Basically, the shorter the path from the switch to a TC is, the less the time it takes for a packet to be delivered. Next, the busyness of a TC is as important as the shortest path because a

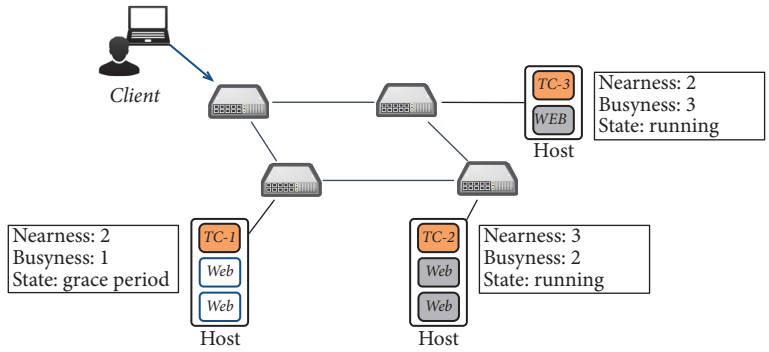

Figure 5: Example scenario of the traffic classifier selection algorithm.

busy TC may impose waiting time on the traffic classification. Last, the state of a TC is also a decision factor. Even if the path is short and a TC is idle, the TC is not available if it is about to recover soon. In other words, a TC in the grace period state cannot be selected:

$$\text{rank} = \text{state} * \left(p * \text{nearness} + (1 - p) * \text{busyness}\right). \quad (8)$$

In order to select the most appropriate TC, the algorithm first computes ranking values for each TC, which is presented in (8). There are three variables: state, nearness, and busyness. The variable *state* can have a binary value meaning whether or not a TC is in the running state. The variable *nearness* is defined as the distance between a TC and a client, and the variable *busyness* denotes how much a TC is busy. The last two variables have relative values. For example, as illustrated in Figure 5, if we assume the TC-3 is the busiest among the 3 TCs, then its busyness value becomes 3. In addition, since the hop count between the client and the TC-2 is the largest, the nearness of TC-2 becomes 3. Finally, the importance of the two variables is regulated by the regulation factor p, which lies between 0 and 1. For example, setting p to 0.5 indicates that we treat the importance of the variables equally.

4.4.4. Central Controller Protection. In Duo, the central controller is an intelligent part that is in charge of controlling server recovery and network flows. These operations are the main mechanisms for Duo to protect a victim system, and thus failure of the central controller can lead to failure of the victim system. However, since the central controller is an SDN application, it is moved to the control plane of SDN, and, as a result, its security is more fortified. Specifically, the central controller is isolated from the outside and thus protected from external threats. This is because communication between the control and data plane in SDN is performed in the OpenFlow protocol [29], and the data plane does not have the capability to affect the control plane. In addition, the central controller is deployed in a distributed fashion along with the SDN controller, and so availability of the central controller is also increased.

5. Performance Evaluation

5.1. Evaluation Environment. In order to evaluate the performance of the proposed system, we have implemented the prototype of Duo in the mininet environment [30]. Mininet allows building SDN prototypes easily, and it is widely used to develop and evaluate OpenFlow applications, providing a realistic network setting. In addition, we use ONOS [31] as the SDN controller, and we build the central controller on top of ONOS using its APIs.

As illustrated in Figure 6, we compose the evaluation topology, which consists of four switches, two clients, and lots of virtual servers including traffic classifiers. To build the servers and the traffic classifiers, we use mininet hosts that are container-like virtual machines. We create a web server that is running on each server, and a copy of Snort is installed on each traffic classifier. Inspired by FlowTags [32], we modify Snort to tag packets if it finds the packets suspicious. In particular, for a realistic performance evaluation, we used real web server logs collected for one year, and the web server is maintained by a software vendor to advertise their products. Since the web server consists of WordPress to publish the contents, we can observe considerable attack trials related to WordPress. Finally, we replay the web logs using Apache JMeter [33].

5.2. Resource Management. In Figure 7, we present the status change of the servers and how they increase or decrease in number. Specifically, there are three boxes in red, which indicate the number of (1) the running servers and (2) additionally required servers to process incoming traffic. If we see the box at the top, the two numbers are the same, which means the volume of traffic remains flat. On the contrary, in the box at the middle, currently there are 4 Gray servers running, and Duo determines that there should be 21 more servers to deal with current traffic. However, due to resource limitation, Duo can add 16 more servers only. Eventually, as shown in the last box, 16 more servers are added, and thus there are 20 Gray servers running.

5.3. Reduced Exposure Time. In this experiment, we compare two ITS architectures and discuss how they are different in

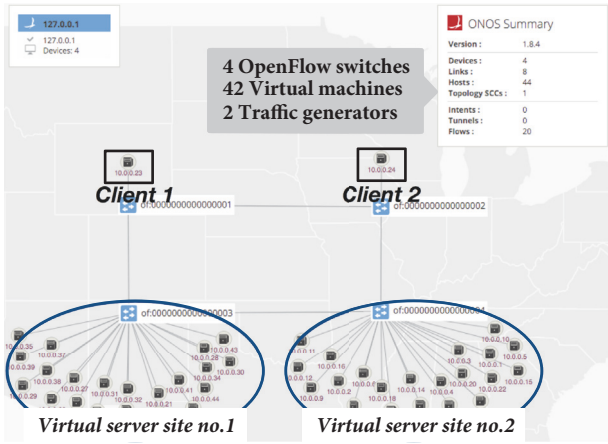

Figure 6: Evaluation environment settings implemented with ONOS.

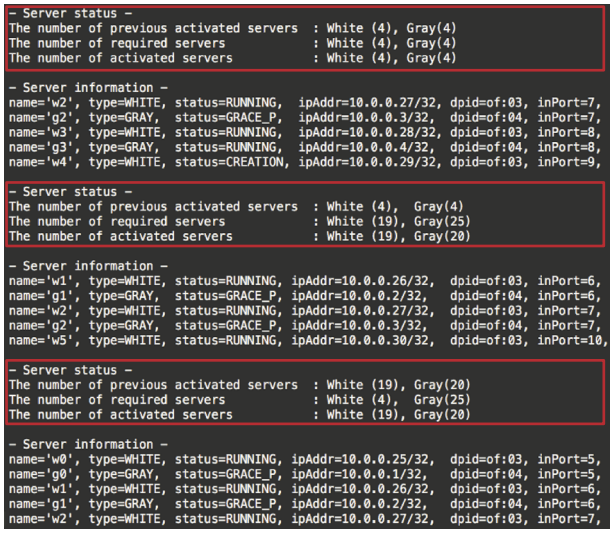

Figure 7: Server status change according to the lifecycle model and server provisioning process.

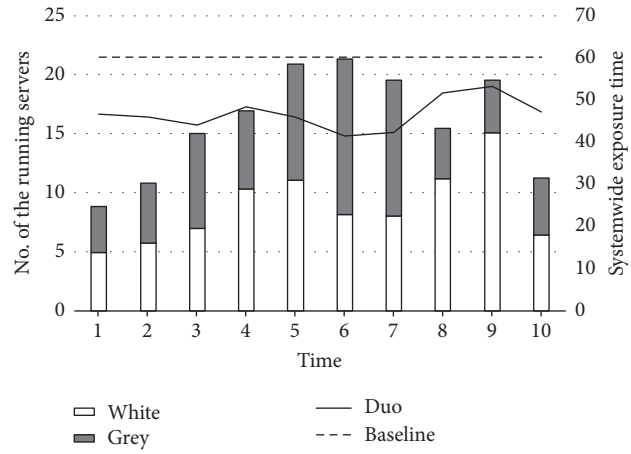

Figure 8: Average exposure time and change in the size of dual server cluster.

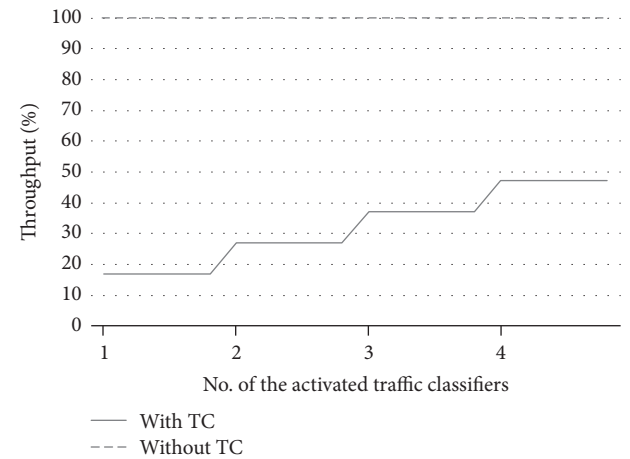

Figure 9: Change in throughput according to the number of the activated traffic classifiers.

terms of response time and exposure time. For this, we built a baseline system that does not classify traffic and employs servers with the same exposure time, E_{Uni}, of 60 s, and then we compare the baseline system and Duo. In case of Duo, we use 60 s and 30 s as E_{White} and E_{Gray}, respectively.

In Figure 8, we show the average exposure time of the baseline system and Duo. According to the volume of two types of traffic, the White and Gray servers are increased and decreased in number. For example, at time point 6, there is more suspicious traffic than benign one, and hence Duo added more Gray servers, which results in decreasing the average exposure time. On the other hand, at time point 9, more benign traffic enters Duo, so White servers are created to handle those traffic.

The average exposure time of Duo, $\overline{E}_{\text{Duo}}$, is always shorter than that of the baseline system, E_{Uni}. Since we set E_{Uni} and E_{White} the same, and E_{Gray} is shorter than them, E_{White} acts as the upper bound of the average exposure time as defined in (1).

5.4. Traffic Classification Overhead. Another concern that needs to be addressed is how Duo can reduce the traffic classification overhead. Since Duo requires incoming network requests to be inspected by a traffic classifier, throughput of the system can be dropped if an insufficient number of traffic classifiers are deployed. To resolve this problem, Duo activates or inactivates traffic classifiers on the fly according to change in the volume of network traffic.

In Figure 9, we illustrate how the additionally activated traffic classifiers can increase throughput of the system. To conduct this experiment, we built a test bed that is composed of multiple container servers on physical machines and SDN-compatible switches, and the test bed has a network topology described in Figure 5. As shown in Figure 9, when compared to the system without traffic classification process, Duo shows about 20% of throughput when a single TC is employed. However, throughput of the system is gradually increased

as extra TCs are activated, and in our settings, the result implies that Duo requires about eight TCs to minimize the classification overhead.

6. Discussion

We now discuss some limitations in our work. First, Duo employs the existing NIDS as the traffic classifier, which is sometimes unable to classify network traffic correctly. For example, if there is a zero-day attack, whose signature is not disclosed yet, then our traffic classifier might tag traffic that the attack generates as benign. However, the exposure time of the White servers could be selectively tuned to be much shorter than usual in order to eliminate such an attack from the servers even if the attack succeeds in infecting them. Thus, Duo can focus more on strengthening the security performance.

Second, in the current design, every network flow is inspected by the TC, which introduces new latency. However, attackers may send benign packets first and attack packets next. So, if we can model the time when attackers start to send suspicious packets, then we are able to send the suspicious packets directly to the Gray servers after inspecting packets only during the period of the modeled time. Extending Duo to reduce the time that network traffic should pass through a traffic classifier is future work.

7. Conclusion

In this paper, we propose Duo, an ITS incorporated in SDN, which is aimed at reducing exposure time by classifying network traffic into two types: suspicious and benign. According to the classification result, suspicious traffic is forwarded to Gray servers and benign one to White servers. Since the Gray servers are more likely to handle malicious traffic, we assign them much shorter exposure time than that of the White servers, which can reduce the average exposure time. To achieve this, we should address two key challenges; one is classification bottleneck problem and the other is server management considering the volume of each type of traffic. To address the first challenge, we employ SND and NFV technologies that enable centralized and adaptive network flow management. With the help of SDN/NFV, we perform the traffic classification in a distributed way. To resolve the second challenge, we formulate the server management problem taking an ILP approach. Our performance evaluation shows that Duo reduces the average exposure time even if the volume of traffic changes. To the best of our knowledge, Duo is the first work that incorporates ITS into SDN.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors thank Jaehyun Nam for his comments on initial design and his help in testing. This work was supported by Institute for Information & Communications Technology Promotion (IITP), a grant funded by Korea government (MSIP) (no. 2016-0-00078, Cloud Based Security Intelligence Technology Development for the Customized Security Service Provisioning).

References

[1] M. Roesch, "Snort - lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX Conference on System Administration LISA '99*, pp. 229–238, USENIX Association, California, Calif, USA, 1999.

[2] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[3] A. Saidane, V. Nicomette, and Y. Deswarte, "The design of a generic intrusion-tolerant architecture for web servers," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 1, pp. 45–58, 2009.

[4] P. Verssimo, N. Neves, and M. Correia, "Intrusion-tolerant architectures: concepts and design," in *Architecting Dependable Systems*, R. D. Lemos, C. Gacek, and A. Romanovsky, Eds., vol. 2677 of *Lecture Notes in Computer Science*, pp. 3–36, Springer Berlin Heidelberg, Berlin, Germany, 2003.

[5] Q. L. Nguyen and A. Sood, "A comparison of intrusion-tolerant system architectures," *IEEE Security & Privacy*, vol. 9, no. 4, pp. 24–31, 2011.

[6] D. Wang, B. B. Madan, and K. S. Trivedi, "Security analysis of sitar intrusion tolerance system," in *Proceedings of the 2003 ACM Workshop on Survivable and Self-regenerative Systems: In Association with 10th ACM Conference on Computer and Communications Security, SSRS '03*, pp. 23–32, ACM, New York, NY, USA, October 2003.

[7] A. K. Bangalore and A. K. Sood, "Securing web servers using self cleansing intrusion tolerance (SCIT)," in *Proceedings of the 2009 2nd International Conference on Dependability, DEPEND 2009*, pp. 60–65, Greece, June 2009.

[8] S. Heo, S. Lee, B. Jang, and H. Yoon, "Designing and implementing a diversity policy for intrusion-tolerant systems," *IEICE Transaction on Information and Systems*, vol. 94-D, no. 4, pp. 1–12, 2016.

[9] B. B. Madan, K. Goševa-Popstojanova, K. Vaidyanathan, and K. S. Trivedi, "A method for modeling and quantifying the security attributes of intrusion tolerant systems," *Performance Evaluation*, vol. 56, no. 1–4, pp. 167–186, 2004.

[10] Q. Nguyen and A. Sood, "Quantitative approach to tuning of a time-based intrusion-tolerant system architecture," in *Proceedings of the 3rd Workshop Recent Advances on Intrusion-Tolerant Systems*, pp. 132–139, 2009.

[11] B. Jang, S. Doo, S. Lee, and H. Yoon, "Hybrid recovery-based intrusion tolerant system for practical cyber-defense," *IEICE Transaction on Information and Systems*, vol. E99D, no. 4, pp. 1081–1091, 2016.

[12] J. Lim, Y. Kim, D. Koo, S. Lee, S. Doo, and H. Yoon, "A novel Adaptive Cluster Transformation (ACT)-based intrusion tolerant architecture for hybrid information technology," *The Journal of Supercomputing*, vol. 66, no. 2, pp. 918–935, 2013.

[13] J. Lim, S. Doo, and H. Yoon, "The design of a robust intrusion tolerance system through advanced adaptive cluster transformation and vulnerability-based VM selection," in *Proceedings of the 2013 IEEE Military Communications Conference, MILCOM '13*, pp. 1422–1428, USA, November 2013.

[14] 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.

[15] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[16] Q. L. Nguyen and A. Sood, "Designing SCIT architecture pattern in a cloud-based environment," in *Proceedings of the Dependable Systems and Networks Workshops (DSN-W), IEEE/IFIP 41st International Conference*, pp. 123–128, IEEE, Hong Kong, June 2011.

[17] 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 20th IEEE International Conference on Network Protocols (ICNP '12)*, pp. 1–6, 2012.

[18] T. Park, Y. Kim, J. Park, H. Suh, B. Hong, and S. Shin, "Qose: quality of security a network security framework with distributed nfv," in *Proceedings of the IEEE International Conference on Communications, ICC '16*, pp. 1–6, May 2016.

[19] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic ddos defense," in *Proceedings of the 24th USENIX Security Symposium (USENIX Security '15)*, pp. 817–832, USENIX Association, Washington, Wash, USA, 2015.

[20] T. Peng, C. Leckie, and K. Ramamohanarao, "Protection from distributed denial of service attacks using history-based IP filtering," in *Proceedings of the 2003 International Conference on Communications (ICC '03)*, vol. 1, pp. 482–486, usa, May 2003.

[21] K. Goseva-Popstojanova, F. Wang, R. Wang et al., "Characterizing intrusion tolerant systems using a state transition model," in *Proceedings of the DARPA Information Survivability Conference and Exposition II, DISCEX '01*, vol. 2, pp. 211–221, 2001.

[22] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: state-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[23] S. Shin, H. Wang, and G. Gu, "A first step toward network security virtualization: from concept to prototype," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 10, pp. 2236–2249, 2015.

[24] The Open Information Security Foundation (OISF), Suricata, https://suricata-ids.org/about.

[25] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: a comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[26] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems SIGMETRICS '05*, pp. 50–60, ACM, New York, NY, USA, 2005.

[27] S. Heo, J. Lim, M. Lee, S. Lee, and H. Yoon, "A novel intrusion tolerant system based on adaptive recovery scheme (ARS)," in *IT Convergence and Security 2012*, K. J. Kim and K.-Y. Chung, Eds., vol. 215 of *Lecture Notes in Electrical Engineering*, pp. 71–78, Springer, Dordrecht, Netherlands, 2013.

[28] Y. Kim, J. Lim, S. Doo, and H. Yoon, "The design of adaptive intrusion tolerant system (ITS) based on historical data," in *Proceedings of the International Conference for Internet Technology and Secured Transactions*, pp. 662–667, December 2012.

[29] Open Networking Foundation, OpenFlow, https://www.opennetworking.org/.

[30] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 19:6, 19:1 pages, ACM, New York, NY, USA, October 2010.

[31] 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, ACM, New York, NY, USA, August 2014.

[32] S. K. Fayazbakhsh, V. Sekar, M. Yu, and J. C. Mogul, "Flowtags: enforcing network-wide policies in the presence of dynamic middlebox actions," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking HotSDN '13*, pp. 19–24, ACM, New York, NY, USA, August 2013.

[33] Apache JMeter, http://jmeter.apache.org.