IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 6, 2017, accepted January 11, 2018, date of publication January 23, 2018, date of current version March 12, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2797214

SecSDN-Cloud: Defeating Vulnerable Attacks Through Secure Software-Defined Networks

IHSAN H. ABDULQADDER[1], DEQING ZOU[1,2], ISRAA T. AZIZ[1,3], BIN YUAN[1], AND WEIMING LI[1]

[1]School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
[2]Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen 518057, China
[3]University of Mosul, Mosul 41002, Iraq

Corresponding author: Weiming Li (lwm@hust.edu.cn)

This work was supported in part by the National Key Research & Development (R&D) Plan of China under Grant 2017YFB0802205, in part by the National Science Foundation of China under Grant 61370230, and in part by the Shenzhen Fundamental Research Program under Grant JCYJ20170413114215614.

ABSTRACT A software-defined network (SDN) is a technology that supports computer network administrators. However, the centralized control plane architecture of SDNs makes them vulnerable to harmful security threats. In this paper, we propose a secure cloud (SecSDN-cloud) architecture that includes user authentication, routing, attack resistance, and third-party monitoring. The goal of this paper is to design an SDN-cloud environment with integrated security that can resist three different attack types: flow table overloading, control plane saturation, and Byzantine attacks. A novel digital signature with chaotic secure hashing is used for user authentication, followed by an enhanced particle swarm optimization multi-class routing protocol to improve the quality of service. Controllers are assigned to switches by integrating an enhanced genetic algorithm with a modified cuckoo search algorithm. The malicious flow identification includes the analysis of five-tuples constructed from features extracted from packets. We implemented the proposed SecSDN-cloud in the OMNeT++ simulator and evaluated its performance in terms of packet loss, end-to-end delay, throughput, latency, and bandwidth.

INDEX TERMS Attackers, cloud computing, routing, security, softwaredefined network.

I. INTRODUCTION

A software-defined network (SDN) focuses on security, energy efficiency and network virtualization to maximize network performance. When an SDN is configured to defend against vulnerable attacks, it must analyze new packets to determine whether they contain threats [1]. Traffic management plays a significant role in minimizing congestion; the OpenFlow protocol is aimed at minimizing packet loss and maximizing the quality of experience (QoE), quality of service (QoS), using resources optimally and evaluating end-to-end network performance [2]–[6]. SDNs are designed to support thousands of applications through manageable network flows and integrated security mechanisms [7]–[9]. An SDN-based cloud computing environment is vulnerable to distributed denial of service (DDoS) attacks. In addition, because several harmful threats may be introduced into the network, SDNs include software-based traffic analysis, and new flows are analyzed [10], [11]. SDN-based cloud architectures are also vulnerable to security threats, including (but not limited to) unauthorized access, data leakage, data modification, forged traffic flows, vulnerabilities in switches/controllers and malicious applications. Threats can be instigated by different individual attackers whose aim is to degrade network performance [12].

A real-time log analysis platform is designed to assess whether the network is overloaded or undergoing an attack [13]. Login management is required to recognize new logins to the network. The security goal is to achieve authenticity, confidentiality and availability by means of data transmission via a single path [14]. Security is enhanced when attackers are detected using probability estimations in the SDN [15]. Media Access Control-Internet Protocol (MAC-IP) mapping is supported to identify attackers by differentiating normal features from attack features. A mismatched MAC address or duplicate IP address can be used to predict an attack. For security purposes, a software-defined system (SDSys) framework is based on the idea of data classification [16]. The problem of high costs is resolved

when using data classifications based on data priority. Packet headers indicate the importance of data and determine the choice of security algorithm. Policy-based approaches are also used to mitigate attacks. Policies are defined as rules that accord with the attributes of flows, switches and hosts [17], [18]. A policy is expressed from the flow ID, source host IP, destination host IP, source MAC, and destination MAC, among others. The defined policies are used to defend against attacks.

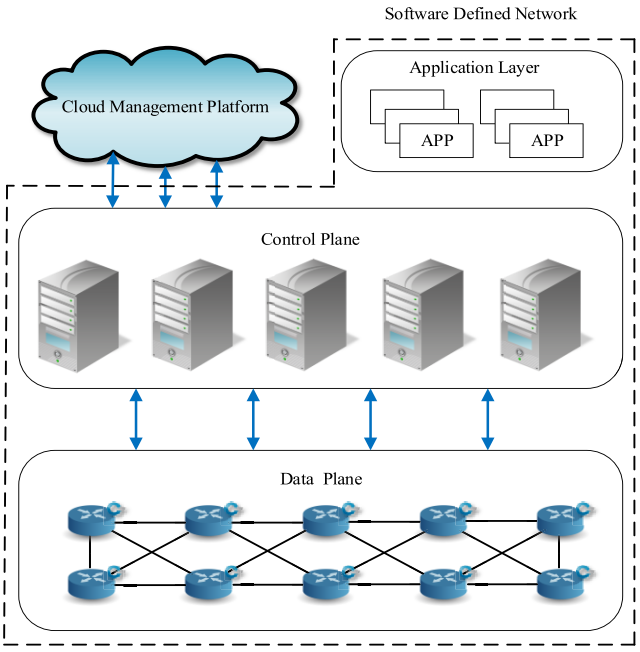


FIGURE 1. Integrated SDN-Cloud model.

An SDN-based cloud environment is designed as shown in Fig. 1. Operational efficiency is enhanced by minimizing the number of hardware components and reducing management requirements. SDN-based clouds focus on adequately addressing various security issues and challenges. Therefore, an SDN-cloud is likely to reduce network latency and defend against multiple harmful attackers.

Brew is a framework designed to perform security policy analysis over a distributed SDN-based cloud environment as described in [18], where all the controllers have knowledge concerning the possible flow rules. Decisions are made based on the actions specified for each flow rule, such as drop, forward and flood. BroFlow is an intrusion detection and prevention system designed similarly to the Bro traffic analyzer [19]. When BroFlow matches a message to the flow table, it initiates corresponding countermeasures at all the switches. Fine-grained policies are defined in a taxonomy-based policy engine for security [20]. A taxonomic relationship expresses rules in which the relationships between users, switches, ports and hosts are well defined. The rules are written based on first-order logic (FOL) in which the identifiers are not fixed. A policy is defined by specifying what network access a specific user is allowed. From the policies, entities enter the SDN network for processing. Firewall policies in SDNs are tested using automatic firewall programs [21]. Model-based floodlight firewall programs are created based on functional requirements and evaluated.

SDNs implement inspections of the security level based on sessions. A controller is enabled with a policy box in a case study in which a handshake message is used for verification [22]. Overall, these SDN features in clouds have been well tested, and their performances have been evaluated in case studies [23].

The major contributions of this work are summarized as follows.

- An integrated, secure SDN and cloud (SecSDN-cloud)-based architecture is designed to resist flow table overloading attacks, control plane saturation attacks and Byzantine attacks.
- The flow table overloading attack is resolved by inspecting packets and their features. A third-party cloud monitors the switches and analyzes the arriving packets.
- The control plane saturation attack is solved using designed multi-controller architecture, which also minimizes SYN-flooding attacks and buffer saturation attacks.
- Byzantine attacks case controller failures, which are solved by using a monitoring server in the network.
- User authentication is provided using a digital signature with a chaotic secure hashing (DS-CSH) algorithm that authenticates users and mitigates the problem of a malicious user entering the network.
- A novel combination of an enhanced genetic algorithm (EGA) and a cuckoo search (CS) is designed to assign controllers based on the available links, latency, controller load balancing and alternative multiple paths.
- An enhanced PSO multi-class (E-PSO) routing protocol is applied to improv the quality of service (QoS) of the communications between nodes. When establishing a route, three significant parameters are considered: node congestion, link congestion and delay.

The remainder of this paper is organized as follows. Section II reviews previous studies on SDN security. Section III describes the proposed SecSDN-cloud environment, including its novel algorithms. Section IV discusses the results obtained from simulations. Finally, Section V provides conclusions and proposes future work.

II. RELATED WORKS

A. CONTROL PLANE SATURATION ATTACK

The decoupled architecture of the control and data planes in an SDN has been subject to a bottleneck in the control plane due to a vulnerability to control plane saturation attacks [24], [25]. SLICOTS was proposed to minimize TCP SYN flooding attacks. In this approach, SLICOTS are fed into controllers to verify connections and ignore or block malicious requests from hosts. The installation of temporary or new rules consumes substantial time and space when

the number of incoming users is large. LineSwitch has also been employed to mitigate this type of attack; it blacklists malicious network traffic. Initially, a connection request from any address is proxied and allowed to enter into the OpenFlow pipeline only after handshake completion. However, involving a connection migration protocol introduces network overhead.

B. BYZANTINE ATTACK

Byzantine attacks are harmful attacks capable of causing problems in any type of network [26], [27]. Multiple controllers for a single device are assigned to solve Byzantine attacks. The Byzantine fault tolerant approach is provided for the state machine replication approach. The algorithm enables assigning switches to multiple controllers, and the switches must wait for all controller replies. Here, the switches are assigned to controllers with a specified capacity, and a maximum of 20 switches are involved in the process. After a random assignment, the next switches are assigned to a controller with another search. Thus, this approach is useful only for smaller, countable switches. Here, consistent services tend to fail when the number of switches is increased.

C. FLOW TABLE OVERLOADING ATTACK

A flow table overloading attack degrades the performance of an SDN [28] by producing new packet processing flows for the switch. Here, a QoS-aware mitigation strategy has been employed to identify idle switches that consists of finding nearby switches with minimum flow table entries. An expiration time is provided for each newer flow that may be either legitimate or malicious. As a result, this approach can mitigate flow table overloading attacks. However, the timeout operation can lead to legitimate flows being denied entry into the network. FloodDefender has been used to secure SDN data and control planes [29]. The FloodDefender architecture includes attack detection, table-miss engineering, packet filtering and flow rule management. The rules can be dynamically adjusted to detect malicious attackers. Flow table management involves both the flow table and cache regions. Newly generated processing flows are stored in the cache region. Although flow table management can improve traffic filtering efficiency, it also increases the space complexity due to the need for a larger cache region.

D. OTHER SECURITY WORKS RELATED TO THE SDN

In [30], AuthFlow was presented to resolve security threats in an SDN. AuthFlow includes a host authentication mechanism, credential-based authentication and a new flow-field-enabled SDN. A host is authenticated by verifying a username and password, an approach that is not secure in a large-scale environment. The security levels were modeled on the design of OpenSec [31], which was designed for flows and to detect malicious traffic. In this approach, flow rules are created in the SDN to improve security and are in a human-readable format. However, it is a straightforward process for attackers to modify these human-readable policies. In [32], a software-defined security (SDSec) controller was designed to mitigate attacks by augmenting controllers with additional functionalities responsible for security. The additional functionalities included two in-memory tables: a host table and a switch table. Maintaining these two tables supported the detection and prevention of attacks. However, although the system identified attacks, it was not able to manage the complex space resulting from the arrival of large numbers of users.

PERM-GUARD was proposed to validate incoming flows from users [33]. In this model, an identity-based signature scheme was introduced to verify flows. New flows are signed by the identity-based signatures to validate flows. Malicious applications can be identified from the flows, but a user with a fake signature may participate in the network because their legitimacy has not been verified. AutoSecSDN was proposed to provide end-to-end security and minimize threats [34]. When any attack is launched against the network, AutoSecSDN verifies network integrity and encrypts traffic. AutoSecSDN is considered a controller; thus, any threats that occur in the network can be predicted only after the controller has participated. Therefore, security in SDN-based cloud remains an active research area focused on solving the associated problems and addressing limitations.

III. PROPOSED SecSDN-CLOUD

A. SYSTEM MODEL

The proposed SecSDN-cloud aims to solve the problems associated with attacks on vulnerabilities in a network. The novel SecSDN-cloud design includes multiple controllers in the control plane, multiple switches in the data plane and a monitoring cloud server, and is intended to mitigate flow table overloading attacks, control plane saturation attacks and Byzantine attacks. Fig. 2 illustrates the proposed SecSDN-cloud architectural model.

The SecSDN-cloud environment is robust against three different attack types initiated by users, and it also supports high-quality communications. The SecSDN-cloud involves three phases to defend against attackers and select a communication route: a user authentication phase, a controller assignment phase and a communication phase. The processes involved in all three phases are discussed in the following sections. In the SecSDN-cloud architecture, a third-party cloud-monitoring server is enabled to identify any malicious flows that enter the network.

Traffic flows from users are analyzed by considering seven significant packet features and analyzing them using five different pre-defined policies. The five actions taken are Alert (analyze the new incoming flow), Quarantine (isolate the new incoming flow), Block (block the flow), Discard (delete the new incoming flow) and Move (install the flow after analysis). The seven significant packet features are the port number, source IP, destination IP, source TCP port, destination TCP port, source Ethernet and destination Ethernet. Based on the switch-controlling policies, malicious flows are

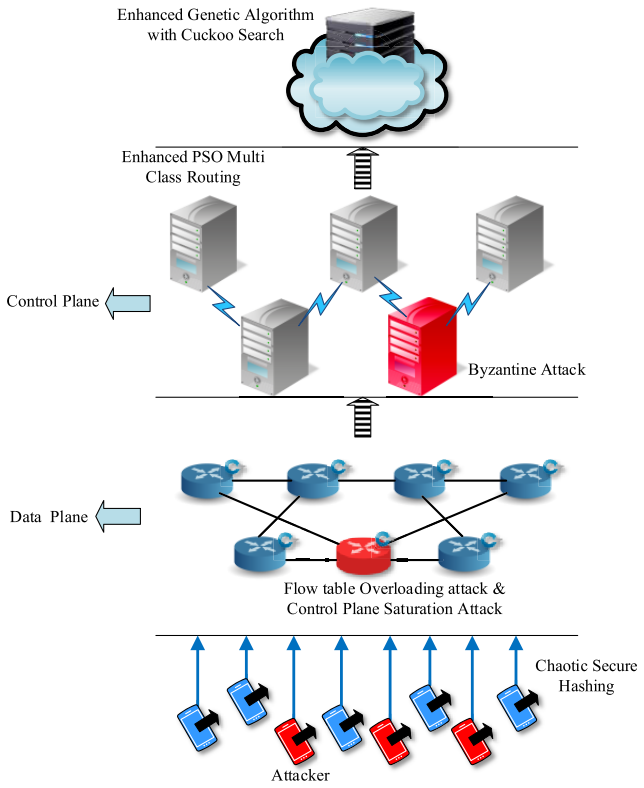


FIGURE 2. SecSDN-cloud environment.

segregated from legitimate flows. All information concerning incoming packets is updated at the monitoring server.

The monitoring server provides both policies for switches participating in the SDN network and trust values whereby switches can validate the legitimacy of each switch. The switch trust values are estimated from the traffic flows. A switch's trust value is defined as the probability of successfully relaying packets and the probability of unused flow table entries. These probabilities are computed using the traffic flows at specific times.

B. USER AUTHENTICATION PHASE

The user authentication phase authenticates each user seeking to participate in the network. The SecSDN-cloud uses the novel DS-CSH algorithm to authenticate users based on the originality of their signatures. The secure hashing algorithm (SHA) is widely used to generate hash functions for security purposes. The SHA achieves integrity, authentication and a digital signature. This study uses SHA3-384, in which "384" denotes the bit length of the digest.

Chaotic hash functions use mapping schemes to determine the trajectory and select valid initial conditions. Chaotic behavior is defined using Lyapunov exponents. When a system is identified as chaotic, the trajectory difference with respect to the initial condition increases according to the time period. The difference is expressed as follows:

$$d_t = d_0 2^{\lambda t}, \tag{1}$$

where d_{00} is the initial distance, d_t is the distance reached at time t and λ defines the Lyapunov exponents, which are formulated by averaging points as follows:

$$\lambda = \frac{1}{t_N - t_0} \sum_{k=1}^{N} \log_2 \frac{d\,(t_k)}{d\,(t_{k-1})}, \tag{2}$$

where N represents the number of points, t_{00} and t_N are the initial time and time at point N, respectively, and $d\,(t_k)$ and $d\,(t_{k-1})$ are the distances at point k and point $k-1$, respectively.

After obtaining the the prediction of λ, the system is considered either chaotic or not chaotic. When $\lambda > 0$, then the system is chaotic. The SHA3 algorithm uses sponge construction in which the message blocks are XORed for conversion into the permutation function f. Considering that the input string (IS) is padded and converted into f, the algorithm performs over bit blocks of width b, rate r and output length d. The constructed sponge is described as follows:

$$Z = Sponge\,[f, pad, r]\,(N, d)\,. \tag{3}$$

From the sponge bit string, we obtain a hash function output of length 384. The use of SHA3-384 does not limit the message size. This hashed function is sent to switches for authentication. When a user is valid, their packets are analyzed based on defined policies. Using hash functions, attackers can be identified before their packets enter the network.

C. CONTROLLER ASSIGNMENT PHASE

The control plane saturation attack type is solved by assigning controllers to switches using the enhanced genetic algorithm combined with the cuckoo search (EGA-CS) algorithm. In this hybrid algorithm, latency and controller load are considered significant metrics. The EGA ranks the chromosomes; then, the CS assigns controllers.

This hybrid algorithm both produces global optimal results and uses minimal computational power. The genetic operators include selection, crossover and mutation. The best "eggs" are chosen using the CS algorithm, where "eggs" denotes the number of controllers participating in the network. The pseudo code for the hybrid EGA-CA algorithm 1 is given below.

1) GENETIC ALGORITHM

The genetic algorithm is designed based on natural biological evaluation processes. This algorithm begins by initializing the population, defined as the number of individuals and represented by chromosomes. The most suitable individual is selected based on the fitness functions. Individuals with higher fitness values are considered to be better solutions than those with lower fitness values. After selecting the individuals, crossover and mutation are applied to the initial population, and a new population is generated. The chromosomes are ranked to find the current best chromosome that has having the best fitness value; then, the ranked chromosomes

Algorithm 1 Pseudo Code for EGA-CS

1. Begin
2. i=0 // Begin genetic algorithm
3. Initialize population P(0)
4. Estimate fitness $F_{GA(i)}$ for P(0)
5. while (t < Max iteration)
   do
   {
   i=i+1
   }
6. Select P(i) among P(i-1)
7. Recombine P(i) with P_c
8. Mutate P(i) with P_m
9. If $\left(F_{GA(i)} > F_{GA(j)}\right)$
   {
   Choose the best fitness
   else
   store value and go to Step 4
   }
   End if
10. End while
11. Rank chromosomes (*Ch*) // End genetic algorithm
12. Classify *Ch* into two groups // Begin cuckoo search
13. For i=1:n
14. Randomly chose i controllers
15. Generate Levy flights solution
16. Evaluate fitness $F_{CS(i)}$
17. If $\left(F_{CS(i)} > F_{CS(j)}\right)$
    {
    Chose the best controller
    else
    Go to Step 14
    }
    End if // End cuckoo search
18. End

are processed with the cuckoo search algorithm to detect the controller that best matches a switch. To mitigate the control plane saturation attach, the identified best controller is assigned to the switches. Becase these types of attacks aim to overload controllers and reduce network performance, the the EGA-CA algorithm 1 chooses the best controller with respect to controller load when assigning controllers. This study estimates fitness in the genetic algorithm based on latency and controller load, which favors high-quality controllers and eliminates low-quality controllers. Chromosomes/individuals represent the controllers involved in the designed SDN network. The algorithm iterates until all the controllers' significant features have been ranked.

2) CS ALGORITHM

The design of the CS algorithm was inspired by a naturally occurring bird species (namely, the cuckoo), and it is a population-based meta-heuristic algorithm defined by the following rules.

- A female cuckoo lays an egg in a nest randomly chosen from among the available nests.
- Future generations follow the best nest with higher quality eggs.
- With a static number of nests, a bird can identify an alien egg at a probability of $P_a \in [0, 1]$. As a result, the bird will either discard the egg or construct a new nest in a new location.

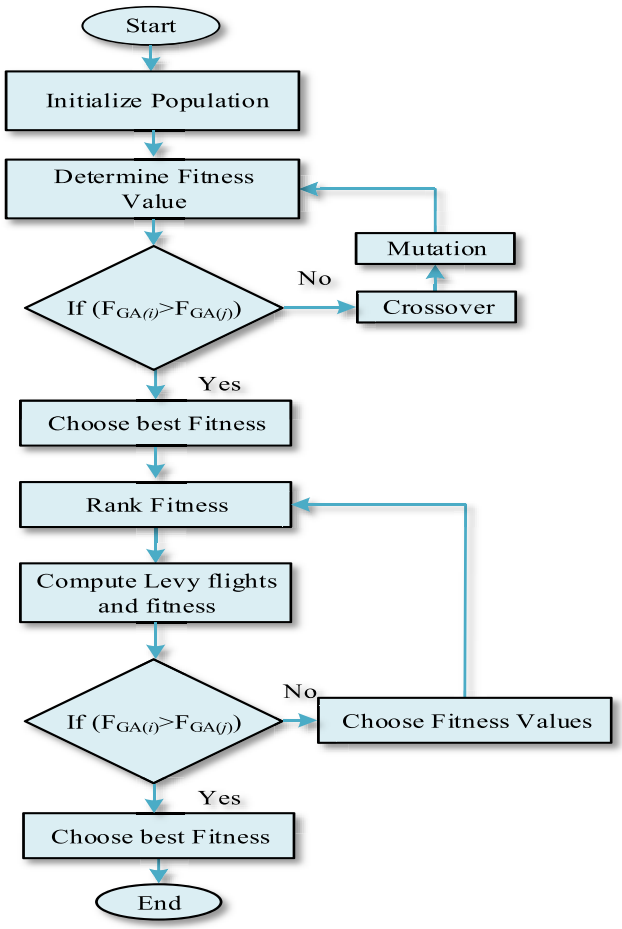


FIGURE 3. Proposed EGA-CS algorithm.

According to the proposed hybrid EGA-CS algorithm shown in Fig. 3, the ranked results from the genetic algorithm are followed by the generation of a new solution that predicts Levy flight as follows:

$$x_i^{(t+1)} = x_i^t + \alpha \oplus Levy\,(\lambda)\,. \tag{4}$$

Here, $\alpha > 0$ is defined as the step size used to update the new solution. After estimating the Levy flight, the fitness is determined from the node links, which are considered a significant metric.

From the evaluated fitness value, the best value is chosen based on the best controller assigned to a switch. The resulting fitness values are ranked according to the previously presented best fitness value. Fig. 3 shows a flow chart for the

hybrid EGA-CS algorithm. Using this algorithm, all packets from the switches are allocated to controllers, and packets do not need to wait to be processed. The constructed network environment is dynamic, i.e., changes occur in the controllers dynamically according to the incoming packets. With respect to time, the number of initialized chromosomes varies; hence, a fitness value will be generated each time. All the fitness values will be ranked, and from this, a single fitness value is chosen randomly to execute the CS algorithm. Here, the randomly chosen fitness value will always be one of the best fitness values obtained. In this work, the minimally loaded controllers are preferentially selected to process the packets arriving from the switches. The controller assignment method is designed to resists control plane saturation attacks.

D. COMMUNICATION PHASE

The communication phase predicts a nodal route for data transmission. The proposed design of this routing protocol is based on the idea of PSO multi-class routing [35]. PSO multi-class routing is conducted through communications among controllers in accordance with the traffic flows. The proposed enhanced PSO multi-class routing protocol supports the link discovery module to aggregate information regarding the network nodes. It then performs route selection, in which a node checks the QoS of other nodes. When another node has a higher QoS value than the current node, then the other node is selected as the one hop neighbor for transmission. The QoS of each node is periodically updated to reflect any changes. In the proposed E-PSO routing protocol, we introduce a data traffic classifier to classify each node; the classifier is used to segregate traffic flows into QoS classes.

The objective function of each controller is achieved by considering the node congestion, link congestion and delay at a node. Each controller classifies the incoming traffic into different classes. The objective function is expressed as follows:

$$OF_n^i = Cn_n^{QoS_i} + \alpha_n \left(Nc^i + Pd^i \right) + \beta_n Lc^i + 1, \quad (5)$$

where $Cn_n^{QoS_i}$ denotes the QoS of a controller, Nc^i is the congestion at a controller, Pd^i is the propagation delay, and Lc^i is the link congestion of a controller. Two other weighted parameters are included, represented as α_n andβ_n. The node congestion is calculated as follows:

$$Nc^i\,(t_c) = Tl^i \times Bo^i. \quad (6)$$

We formulate the current congestion on the controller $Nc^i\,(t_c)$ in terms of ρ^i (i.e. traffic load) and ω^i (i.e. buffer occupancy). These metrics are formulated as follows:

$$Tl^i = \frac{\lambda^i}{\mu^i} \quad (7)$$

$$Bo^i = \frac{Q^i}{L^i}, \quad (8)$$

where λ^i is the arrival rate of controller$i$, $\mu^i$is the service rate of controller$i$, and Q^i and L^i are the queue length and buffer length of controller$i$, respectively. Then, the Pd^i of the ith controller is formulated using

$$Pd^i = \frac{D^{ij}}{Tx_{rt}^{ij}}, \quad (9)$$

where D^{ij} is the distance between controllers i and$j$ and Tx_{Sd} denotes the transmission rate from one controller to another. Therefore, minimizing Nc^i, Lc^i and Pd^i leads to an increase in the QoS. Each controller classifies the incoming traffic into different classes and then routes the data packets for processing.

Therefore, the overall design and processing of the proposed SecSDN-cloud environment resists three different attack types. Multiple controllers, multiple switches and a monitoring server are involved to mitigate the harm from an attacker's activities.

IV. RESULTS EVALUATION

In this section, we evaluate the proposed SecSDN-cloud design with novel algorithms designed for security. This section comprises three sub-sections: simulation setup, attack analysis and comparative analysis.

A. SIMULATION SETUP

The proposed novel SecSDN-cloud architecture is implemented in the OMNeT++ simulation environment initially developed in Germany in 2006 by OpenSin Ltd. . OMNeT++ was selected due to its strong support for SDNs because it includes the OpenFlow inet file and other SDN supported libraries.

TABLE 1. Simulation parameters.

SPECIFICATIONS	Values
Number of hosts	16
Number of controllers	3
Number of switches	4
Queue type	Drop tail queue
Buffer capacity	10
Data rate	100 Mbps
Send interval	2 seconds
Simulation time	30 seconds

Table 1 illustrates the most significant parameters used in the proposed algorithm simulations. We vary the number of hosts, OpenFlow switches and controllers to analyze the network performance. However, the parameters are not limited to this list.

B. ATTACK ANALYSIS

The main goal of the proposed work is to resist flow table overloading attacks, control plane saturation attacks and Byzantine attacks.

1) FLOW TABLE OVERLOADING ATTACK

In the SDN, each switch that participates in the data plane is associated with a specified flow table size. Because the

flow table size is constrained, switches are vulnerable to attacks that attempt to overload the flow table. An overloaded flow table at a switch with arrived packets from users causes longer packet processing times. Thus, this type of attack can be changed into a DDoS attack. In the SecSDN-cloud framework, a third-party monitoring server provides policies to the switches. The switches take action based on the policies to avoid malicious flows. Thus, even when a switch's flow table is completely occupied, the flows are migrated based on trust values. Therefore, the SecSDN-cloud approach resists flow table overloading attacks.

2) CONTROL PLANE SATURATION ATTACK

The communications necessary between decoupled control and data planes in SDNs can result in a bottleneck that increases the load at the controllers. This condition can be induced by control plane saturation attacks, which are also known as buffer saturation attacks. This type of attack can also be caused by SYN flooding attacks. The SecSDN-cloud architecture involves multiple controllers, which minimizes loads and resists control plane saturation attacks. By using multiple controllers, the load can be split among controllers, consequently, no single controller is fully loaded.

3) BYZANTINE ATTACK

Byzantine attacks cause controller failures that lead to packet processing delays. To guard against this attack type, SecSDN-cloud employs a third-party cloud-monitoring server to execute the EGA-CS algorithm that assigns controllers to switches. This approach to controller assignment mitigates Byzantine attacks.

C. COMPARATIVE ANALYSIS

In this sub-section, seven comparative plots are shown to analyze the ability of the proposed SecSDN-cloud environment to resist attacks. The parameters that are considered are saturation time, bandwidth consumption, connectivity loss, holding time, packet loss, throughput and end-to-end delay. The last three parameters illustrate the proposed framework's QoS support during data transmissions.

1) AVERAGE BUFFER SATURATION TIME

Control plane saturation attacks occur due to controller buffer overloads. A faster-filling buffer corresponds to a more intense attack.

Fig. 4 depicts the results obtained from the proposed simulation for the SecSDN-cloud environment by plotting the average saturation time of buffer overload with respect to the attacker's bandwidth. In LineSwitch, controller buffers fill faster than in the proposed SecSDN-cloud because SecSDN-cloud involves multiple controllers. In our approach, the size of a controller's buffer is immaterial in ensuring security against this saturation attack type. Although buffer sizes can be increased, an increased attack intensity can easily outpace increased buffer sizes. Therefore, the proposed SecSDN-cloud can withstand an approximately 10% higher saturation time than can LineSwitch.

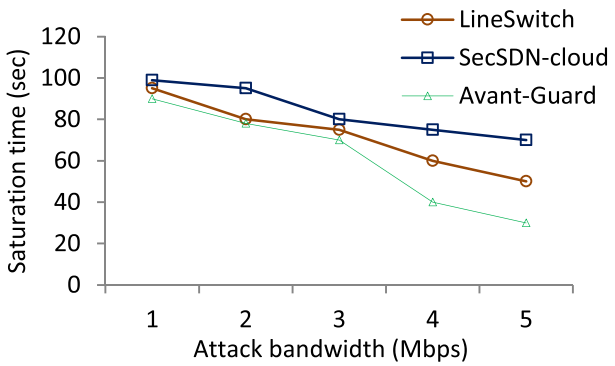


FIGURE 4. Comparison for control plane saturation attack.

2) NETWORK CONNECTIVITY LOSS

Connectivity loss is also a significant metric when studying the security achieved by the SecSDN-cloud. Connectivity loss is defined as the loss of connectivity between two entities, which degrades network performance.

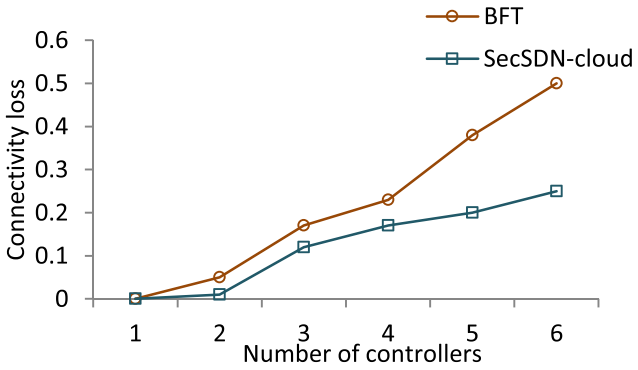


FIGURE 5. Comparison of the response to a Byzantine attack.

Fig. 5 illustrates connectivity loss based on the level of security during a Byzantine attack. Assigning controllers to switches supports the connectivity between the data and control planes. Therefore, the connectivity loss is detectably smaller in SecSDN-cloud than in the BFT. As the number of controllers increases, the connectivity loss also shows a gradual increase but its connectivity loss is never higher than that of the BFT.

3) HOLDING TIME

To measure the effectiveness of flow table overloading attacks on the proposed SecSDN-cloud, we consider holding time to be a significant metric. When the holding time declines, the likelihood of an attack is higher. An architecture that demonstrates longer holding times even under higher attack rates reflects an SDN that provides greater security.

The efficacy of flow table overloading attacks is demonstrated in Fig. 6. The attackers target a single switch to occupy resources and prevent legitimate packets from entering the switch. A flow table with limited resources has difficulty withstanding a flow table overloading attack. In the proposed

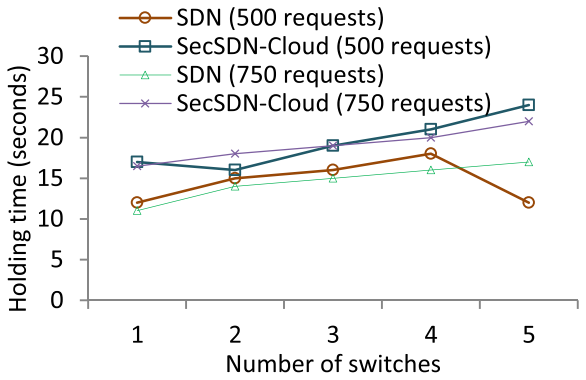


FIGURE 6. Comparison of the robustness against flow table overloading attacks.

SecSDN-cloud, the incoming packets are analyzed with respect to policies provided by the server. Thereby, all malicious traffic is blocked, and the holding time increases.

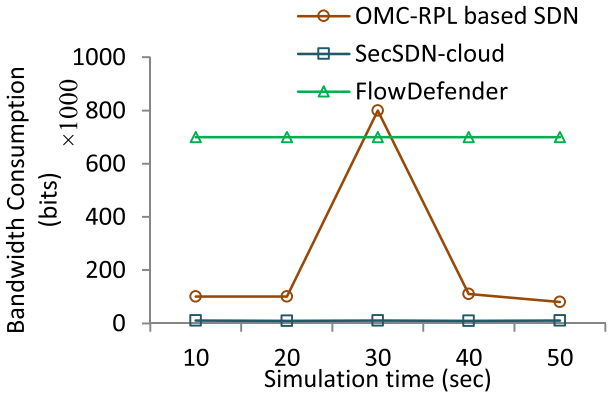


FIGURE 7. Comparison of bandwidth consumption.

4) BANDWIDTH CONSUMPTION

Bandwidth consumption is associated with the increased participation of attackers in a network. Although energy is not a limiting constraint in SDNs, the controllers should consume minimal bandwidth when processing flows. Fig. 7 shows a comparison of bandwidth consumption in existing SDNs and the proposed SecSDN-cloud approach. Maintaining a centralized controller consumes more bandwidth; thus, the SecSDN-cloud uses multiple distributed controllers. In addition, it resists attackers by allowing only legitimate flows into the network. Therefore, legitimate traffic processing does not consume additional bandwidth in the SecSDN cloud.

5) EFFECTIVENESS OF THE QoS

The QoS represents the ability of a network to provide higher-quality services to specific users. This metric is measured in relation to various parameters such as bit rate, throughput, packet loss and delay.

In this work, the QoS effectiveness is defined in terms of three significant throughput parameters as well as end-to-end delay and packet loss.

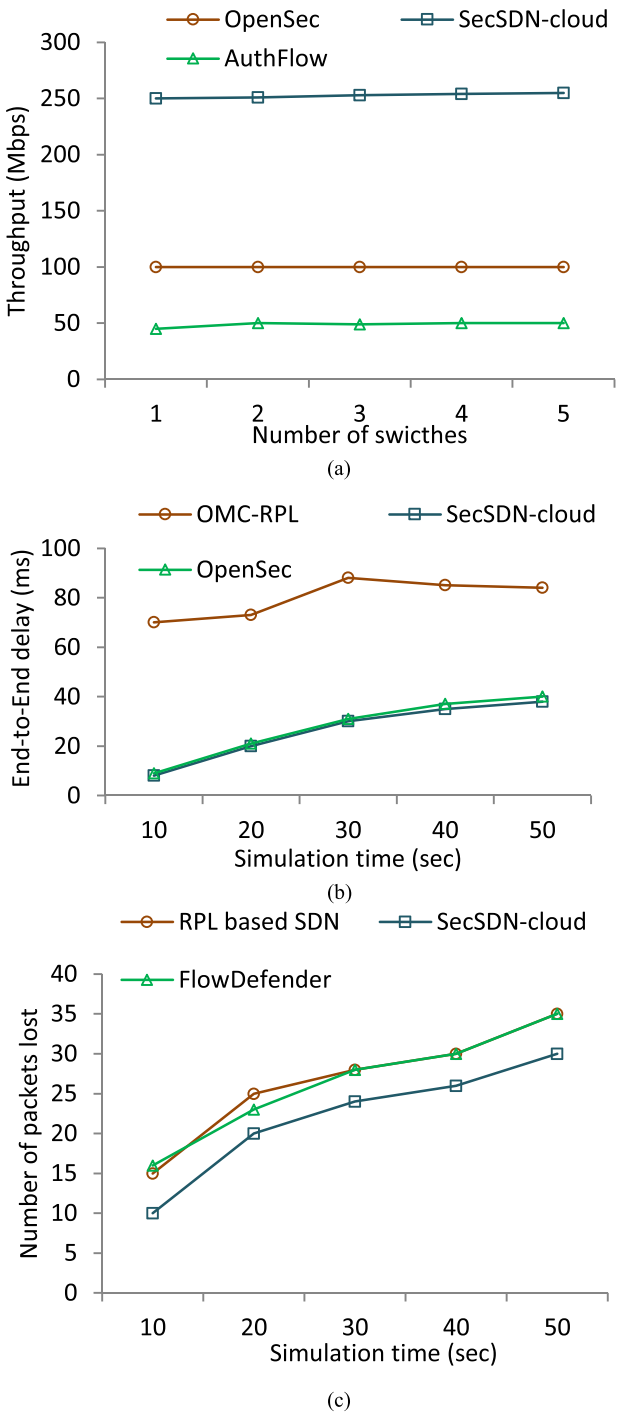


FIGURE 8. (a) Throughput. (b) End-to-end delay. (c) Packet loss.

Network throughput is defined as the amount of data transmitted from one entity to another in a specified time period. The throughput, T_p, is calculated using the following formula:

$$T_p = \frac{NP_s}{T}, \tag{10}$$

where T is the time interval of transmission and NP_s is the number of packets transferred. Fig. 8(a) shows improvements in throughput relative to the proposed

SecSDN-cloud architecture. Defending against attackers is the major reason for the improvements in throughput.

Figure 8 (b) depicts the results obtained for the end-to-end delay, which is defined as the time required to transmit a packet from a source entity to a destination entity across the network. In the SecSDN-cloud, this delay increases gradually as the simulation time increases because the number of incoming packets is increased. Delay occurs because of node failures and connectivity losses. The SecSDN-cloud resolves node failure caused by attackers and sustains connectivity. Thus, improvements to these parameters are reflected by a reduction of the end-to-end delay metric. The proposed novel algorithms effectively achieve improvements in QoS.

Packet loss results from packets discarded by a device intended to accept the packet due to overloading. Because buffer and queue sizes are limited, packet overloading occurs when the system is unable to accept packets that exceed the existing level. Packets stemming from network attacks can cause packet losses for legitimate users. Figure 8 (c) shows that the SecSDN-cloud minimizes packet losses because it can mitigate harmful attacks.

As demonstrated through the simulations and analyses, the design of the proposed SecSDN-cloud is superior to that of previous SDNs. The proposed SecSDN-cloud includes defenses against harmful attacks and can be applied in conjunction with upcoming novel applications.

V. CONCLUSION

We propose a SecSDN-cloud architecture designed to defend against three different attack types (flow table overloading attacks, control plane saturation attacks and Byzantine attacks). The need for user authentication is addressed through the DS-CSH, which uses SHA3-384. Only after validating a user do that user's packets proceed for analysis using the established policies. Policies are defined by the monitoring cloud server in terms of packet features. This approach serves to block malicious traffic and mitigate flow table overloading attacks. To resist Byzantine attacks, switches are assigned to controllers using a novel EGA-CS algorithm. Packets that reach the control plane perform an enhanced form of the PSO multiclass routing protocol to provide a higher QoS. Thus, the proposed SecSDN-cloud provides a secure SDN environment with a higher QoS that can support more users.

In the future, we plan to apply this proposed SecSDN-cloud environment to a 5G network that supports higher data rates and coverage.

REFERENCES

[1] D. B. Rawat and S. R. Reddy, "Software defined networking architecture, security and energy efficiency: A survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 325–346, 1st Quart., 2017.

[2] A. Mendiola, J. Astorga, E. Jacob, and M. Higuero, "A survey on the contributions of software-defined networking to traffic engineering," *IEEE Commun. Surveys Tuts.*, vol. 198, no. 2, pp. 918–953, 2nd Quart., 2017.

[3] J. M. Wang, Y. Wang, X. Dai, and B. Bensaou, "SDN-based multi-class QoS guarantee in inter-data center communications," *IEEE Trans. Cloud Comput.*, to be published.

[4] J. Liu, Y. Li, H. Song, and D. Jin, "NetWatch: End-to-end network performance measurement as a service for cloud," *IEEE Trans. Cloud Comput.*, to be published.

[5] K. Sood, S. Yu, Y. Xiang, and H. Cheng, "A general QoS aware flow-balancing and resource management scheme in distributed software-defined networks," *IEEE Access*, vol. 4, pp. 7176–7185, 2016.

[6] K. Xu, H. Xiong, C. Wu, D. Stefan, and D. Yao, "Data-provenance verification for secure hosts," *IEEE Trans. Depend. Sec. Comput.*, vol. 9, no. 2, pp. 173–183, Mar./Apr. 2012.

[7] M. H. Kabir, "Software defined networking (SDN): A revolution in computer network," *J. Comput. Eng.*, vol. 15, no. 5, pp. 103–106, 2013.

[8] C. Chen, X. Hu, K. Zheng, X. Wang, Y. Xiang, and J. Li, "HBD: Towards efficient reactive rule dispatching in software-defined networks," *Tsinghua Sci. Technol.*, vol. 21, no. 2, pp. 196–209, 2016.

[9] H. Jin, W. Dai, and D. Zou, "Theory and methodology of research on cloud security," *Sci. China Inf. Sci.*, vol. 59, no. 5, pp. 050105-1–050105-3, 2016.

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

[11] T. Xu, D. Gao, P. Dong, H. Zhang, C. H. Foh, and H.-C. Chao, "Defending against new-flow attack in SDN-based Internet of Things," *IEEE Access*, vol. 5, pp. 3431–3443, 2017.

[12] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2317–2346, 4th Quart., 2015.

[13] B. Siniarski, C. Olariu, P. Perry, T. Parsons, and J. Murphy, "Real-time monitoring of SDN networks using non-invasive cloud-based logging platforms," in *Proc. IEEE 27th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Sep. 2016, pp. 1–6.

[14] S. Rass, B. Rainer, M. Vavti, J. Göllner, A. Peer, and S. Schauer, "Secure communication over software-defined networks," *Mobile Netw. Appl.*, vol. 20, no. 1, pp. 105–110, 2015.

[15] H. Ma, H. Ding, Y. Yang, Z. Mi, and M. Zhang, "SDN-based ARP attack detection for cloud centers," in *Proc. IEEE 12th Int. Conf. Auto. Trusted Comput., Ubiquitous Intell. Comput., IEEE 15th Int. Conf. Scalable Comput. Commun. Assoc. Workshops (UIC-ATC-ScalCom)*, Aug. 2015, pp. 1049–1054.

[16] Y. Jararweh, M. Al-Ayyoub, and H. Song, "Software-defined systems support for secure cloud computing based on data classification," *Ann. Telecommun.*, vol. 72, nos. 5–6, pp. 335–345, 2017.

[17] K. K. Karmakar, V. Varadharajan, and U. Tupakula, "Mitigating attacks in software defined network (SDN)," in *Proc. 4th Int. Conf. Softw. Defined Syst. (SDS)*, 2017, pp. 112–117.

[18] S. Pisharody, J. Natarajan, A. Chowdhary, A. Alshalan, and D. Huang, "Brew: A security policy analysis framework for distributed SDN-based cloud environments," *IEEE Trans. Depend. Sec. Comput.*, to be published.

[19] M. A. Lopez, D. M. F. Mattos, and O. C. M. Duarte, "An elastic intrusion detection system for software networks," *Ann. Telecommun.*, vol. 71, nos. 11–12, pp. 595–605, 2016.

[20] C. Banse and J. Schuette, "A taxonomy-based approach for security in software-defined networking," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[21] I. Alsmadi, M. Munakami, and D. Xu, "Model-based testing of SDN firewalls: A case study," in *Proc. 2nd Int. Conf. Trustworthy Syst. Their Appl. (TSA)*, 2015, pp. 81–88.

[22] A. Ranjbar, M. Komu, P. Salmela, and T. Aura, "An SDN-based approach to enhance the end-to-end security: SSL/TLS case study," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Apr. 2016, pp. 281–288.

[23] V. Shamugam, I. Murray, J. Leong, and A. S. Sidhu, "Software defined networking challenges and future direction: A case study of implementing SDN features on OpenStack private cloud," in *Proc. IOP Conf. Ser., Mater. Sci. Eng.*, 2016, p. 012003.

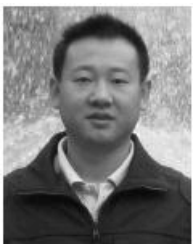
[24] R. Mohammadi, R. Javidan, and M. Conti, "SLICOTS: An SDN-based lightweight countermeasure for TCP SYN flooding attacks," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 2, pp. 487–497, Jun. 2017.

[25] M. Ambrosin, M. Conti, F. De Gaspari, and R. Poovendran, "Lineswitch: Tackling control plane saturation attacks in software-defined networking," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 1206–1219, Apr. 2017.

[26] R. Cao, T. F. Wong, T. Lv, H. Gao, and S. Yang, "Detecting Byzantine attacks without clean reference," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2717–2731, Dec. 2016.

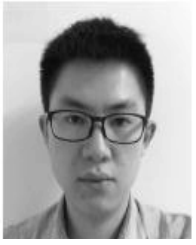
[27] H. Li, P. Li, S. Guo, and A. Nayak, "Byzantine-resilient secure software-defined networks with multiple controllers in cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 4, pp. 436–447, Oct. 2014.
[28] B. Yuan, D. Zou, S. Yu, H. Jin, W. Qiang, and J. Shen, "Defending against flow table overloading attack in software-defined networks," *IEEE Trans. Services Comput.*, to be published.
[29] S. Gao, Z. Peng, B. Xiao, A. Hu, and K. Ren, "FloodDefender: Protecting data and control plane resources under SDN-aimed DoS attacks," in *Proc. IEEE INFOCOM*, May 2017, pp. 1–9.
[30] D. M. F. Mattos and O. C. M. B. Duarte, "AuthFlow: Authentication and access control mechanism for software defined networking," *Ann. Telecommun.*, vol. 71, nos. 11–12, pp. 607–615, 2016.
[31] A. Lara and B. Ramamurthy, "OpenSec: Policy-based security using software-defined networking," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 1, pp. 30–42, Mar. 2016.
[32] M. Al-Zewairi, D. Suleiman, and S. Almajali, "An experimental software defined security controller for software defined network," in *Proc. 4th Int. Conf. Softw. Defined Syst. (SDS)*, 2017, pp. 32–36.
[33] M. Wang, J. Liu, J. Chen, X. Liu, and J. Mao, "Perm-guard: Authenticating the validity of flow rules in software defined networking," *J. Signal Process. Syst.*, vol. 86, nos. 2–3, pp. 157–173, 2017.
[34] R. Khondoker, P. Larbig, D. Senf, K. Bayarou, and N. Gruschka, "AutoSecSDNDemo: Demonstration of automated end-to-end security in software-defined networks," in *Proc. IEEE NetSoft Conf. Workshops (NetSoft)*, Jun. 2016, pp. 347–348.
[35] M. Alishahi, M. H. Y. Moghaddam, and H. R. Pourreza, "Multi-class routing protocol using virtualization and SDN-enabled architecture for smart grid," in *Proc. Peer-to-Peer Netw. Appl.*, Dec. 2016, pp. 1–17.

IHSAN H. ABDULQADDER received the B.Sc. degree in electrical engineering from Baghdad University, Baghdad, Iraq, in 2004, and the M.Sc. degree in computer science—computers, information, and network security from DePaul University, Chicago, IL, USA, in 2010. He is currently pursuing the Ph.D. degree in computer science with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. He has worked in many positions such as on network switching subsystems, as a Consultant, a Core Network Engineer, a Trainer, and as a Faculty Member with the Computer Science Department. His research interests include security in SDN, NFV, and cloud computing.



DEQING ZOU received the Ph.D. degree from the Huazhong University of Science and Technology (HUST) in 2004. He is currently a Professor of computer science with HUST. His main research interests include system security, trusted computing, virtualization, and cloud security. He was the leader of one 863 project in China and of three National Natural Science Foundation of China projects and has been a core member of several important national projects, such as the National 973 Basic Research Program of China. He has applied for almost 20 patents, published two books—one is *Xen Virtualization Technologies* (Huazhong University of Science and Technology Press, 2009) and the other is *Trusted Computing Technologies and Principles* (Science Press, 2011)—and over 50 papers in the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the IEEE Symposium on Reliable Distributed Systems, and many other publications. He has continually served as a Reviewer for several prestigious journals, including the IEEE TPDS, the IEEE TOC, the IEEE TDSC, the IEEE TCC, and others. He is on the Editorial Boards of four international journals and has served as a PC Chair or PC member on over 40 international conferences.

ISRAA T. AZIZ received the B.Tech. degree in computer engineering from the Mosul Technical College, Mosul, Iraq, in 2007, and the M.Tech degree in computer engineering from the Sam Higginbottom University of Agriculture, Technology and Sciences, Allahabad, India, in 2014. She is currently pursuing the Ph.D. degree in computer science with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. She was a Lecturer with the University of Mosul, Mosul, Iraq. Her research interests include computer security.



BIN YUAN received the B.S. degree in computer science and technology from the Huazhong University of Science and Technology, Wuhan, China, in 2013, where he is currently pursuing the Ph.D. degree. His research interests include security in SDN and cloud computing.

WEIMING LI received the Ph.D. degree in computer science from the Huazhong University of Science and Technology, China, in 2006. He is currently an Associate Professor with the Network and Computing Center, Huazhong University of Science and Technology. His current research focuses on system security, especially malware analysis and detection using binary analysis techniques. He also has interests in network security. He has published over 30 refereed papers and won the first prize of the Hubei province science and technology progress in 2011.