Computer Networks 81 (2015) 308–319

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

DDoS attack protection in the era of cloud computing and Software-Defined Networking

Bing Wang *, Yao Zheng, Wenjing Lou, Y. Thomas Hou

Virginia Polytechnic Institute and State University, Blacksburg, VA, USA

ARTICLE INFO

Article history:
Received 16 September 2014
Received in revised form 27 January 2015
Accepted 26 February 2015
Available online 6 March 2015

Keywords:
DDoS mitigation
Software-Defined Networking
Graphical model

ABSTRACT

Cloud computing has become the real trend of enterprise IT service model that offers cost-effective and scalable processing. Meanwhile, Software-Defined Networking (SDN) is gaining popularity in enterprise networks for flexibility in network management service and reduced operational cost. There seems a trend for the two technologies to go hand-in-hand in providing an enterprise's IT services. However, the new challenges brought by the marriage of cloud computing and SDN, particularly the implications on enterprise network security, have not been well understood. This paper sets to address this important problem.

We start by examining the security impact, in particular, the impact on DDoS attack defense mechanisms, in an enterprise network where both technologies are adopted. We find that SDN technology can actually help enterprises to defend against DDoS attacks if the defense architecture is designed properly. To that end, we propose a DDoS attack mitigation architecture that integrates a highly programmable network monitoring to enable attack detection and a flexible control structure to allow fast and specific attack reaction. To cope with the new architecture, we propose a graphic model based attack detection system that can deal with the dataset shift problem. The simulation results show that our architecture can effectively and efficiently address the security challenges brought by the new network paradigm and our attack detection system can effectively report various attacks using real-world network traffic.

1. Introduction

As cloud computing provides on-demand, elastic, and accessible computing services, more and more enterprises begin to embrace this paradigm shift by moving their database and applications into the cloud. At the same time, another epochal concept of the Internet architecture comes to forefront, i.e., Software-Defined Networking (SDN). While cloud computing facilitates the management of computation and storage resources, SDN is proposed to address another laborious issue hindering the evolvement of today's Internet, i.e., the complicated network management. Besides the fact that SDN has been proposed as a candidate of the next generation Internet architecture, companies like Google have already adopted SDN in their internal data centers. Thus, the arrival of the era when cloud computing and SDN go hand-in-hand in providing enterprise IT services is looming on the horizon.

Besides all the widely perceived benefits, the marriage between cloud computing and SDN may also introduce potential risks, especially on network security. Among all the network security problems, we first take a look at Denial-of-Service (DoS) attack. A DoS attack and its distributed version, Distributed Denial-of-Service (DDoS) attack, attempt to make a service unavailable to its intended users by draining the system or network

* Corresponding author.
E-mail addresses: bingwang@vt.edu (B. Wang), zhengyao@vt.edu (Y. Zheng), wjlou@vt.edu (W. Lou), thou@vt.edu (Y.T. Hou).

http://dx.doi.org/10.1016/j.comnet.2015.02.026
1389-1286/

resource. Although network security experts have been devoting great efforts for decades to address this issue, DDoS attacks continue to grow in frequency and have more impact recently. Existing DDoS attack defense solutions (to list a few [1–4]) assume a fully controlled network by the network administrators of enterprises. Therefore, the network administrators could place certain hardware pieces in the network to detect or mitigate DDoS attacks. However, in the new network paradigm of cloud computing and SDN, these assumptions no longer stand. Other researchers [5,6] focus on exploiting the benefits of cloud or SDN to defend DDoS attacks. But their target victims still reside in the traditional network environment, which makes their solutions unsuitable for the new network paradigm. To the best of our knowledge, little effort in research community has been made to look into the potential problems or opportunities to defend DDoS attacks in the new enterprise network environment that adopts both cloud computing and SDN.

In this paper, we first analyze the impact of the combination of cloud computing and SDN on DDoS attack defense. We discuss the potential issues under this new paradigm as well as opportunities of defending DDoS attacks. Based on our analysis, we claim that if designed properly, SDN can actually be exploited to address the security challenges brought by cloud computing and the DDoS attack defense can be made more effective and efficient in the era of cloud computing and SDN. We then propose a new DDoS attack mitigation architecture using software-defined networking (abbreviated as DaMask) to demonstrate and substantiate our findings. DaMask contains two modules: an anomaly-based attack detection module DaMask-D, and an attack mitigation module DaMask-M. We build our DaMask-D module based on a graphical probabilistic inference model. Compared with existing graphical model based detection schemes [7–9] which only have model training and testing phases, our DaMask-D features an additional model updating phase to address the *dataset shift* problem in the real world. The dataset shift refers to the fact that the network traffic conditions when we build the model differ from the actual traffic conditions when we use the model. This fact varies from the common assumption used in the existing works where the attack patterns learned from the training data are assumed to be no different from the attack patterns in the future. Our contributions can be summarized as follows:

1. To the best of our knowledge, we are among the first to bring the attention of the impact on DDoS attack defense of the new network paradigm, which is a combination of cloud computing and SDN. Based on our analysis, we find that the marriage of SDN and cloud computing provides a unique opportunity to enhance the DDoS attack defense in an enterprise network environment.
2. To substantiate our claim, we propose DaMask, a highly scalable and flexible DDoS attack mitigation architecture that exploits SDN technique to address the new security challenges brought by cloud computing, including the extended defense perimeter and the dynamic network topological changes.
3. Our DaMask-D module in the DaMask architecture features an additional model update phase, compared to existing graphical-model based network attack detection schemes, which successfully handles the dataset shift problem in the real world and achieves a higher detection rate.
4. At last, we implement our proposed structure and performed a simulation based evaluation using the Amazon EC2 cloud service. The results show that our scheme works well under the new network paradigm and incurs limited computation and communication overhead, which is a crucial requirement of DDoS protection in cloud computing and SDN.

Compared with our preliminary NPSec work [10] which presented the DaMask framework, the journal version completes the DDoS attack defense solution by including an attack detection system in Section 4. The attack detection system which is based on the graphical model detection is not only tailored to accommodate the unique requirement of DDoS attack defending in cloud computing, but also manages to address the data shift problem which decreases the detection performance in most machine learning based solutions. We also add performance evaluation results of the detection module in Section 5.3 including the performance of detecting attacks and the ability of adapting the data shift issue. We organize the remainder of the paper as follows. We analyze the impact of cloud computing and SDN on DDoS attack defense in Section 2. Based on our analysis, we formulate the problem and present our DaMask architecture design in Section 3. The technical details of the DaMask-D module is discussed in Section 4. Section 5 presents the simulation setting and the results. Related work are reviewed and compared with our work in Section 6. We draw concluding remarks in Section 7.

2. Analysis

In this section, we briefly review cloud computing and SDN. Then we analyze the impact of the combined technologies on the network protection against DDoS attacks.

2.1. Cloud computing

Cloud computing is a computing model which manages a pool of configurable computing resources. Cloud computing can be categorized as *public cloud, private cloud* and *hybrid cloud* in terms of deployment. While public cloud and private cloud are used by public and a single organization, respectively, hybrid cloud is a composition of public and private cloud infrastructures. As a result, hybrid cloud share the properties of both public cloud and private cloud. Hybrid cloud allows companies keeping their critical applications and data in private while outsourcing others to public. Thus, we focus on analyzing the impact of hybrid cloud on DDoS attack defense.

2.2. Impact of cloud computing on DDoS attack defense

Nowadays, attackers can launch various DDoS attacks including resource-focused ones (e.g. network bandwidth, memory, and CPU) and application-focused ones (e.g. web applications, database service) from almost everywhere. To be realistic, we have to assume attackers can reside either in a private network, in a public network, or in both. To this end, we find the following properties of cloud computing affect DDoS attack defense.

1. Instead of users, cloud providers control network and computation resources, i.e., physical servers. This property differs from the system model in the traditional DDoS attack defense, where the protected application servers are within the defender controlled network.
2. Resource allocation and virtual machine migration are new sources of network topological changes from the defender's view. Moreover, the resource allocation and virtual machine migrations processes are fast-paced. The DDoS attack defense must be able to adapt to a dynamic network with frequent topological changes and still maintain high detection rate and prompt reaction capability.
3. All cloud users share the same network infrastructure of the cloud. This raises a reliable network separation requirement, which has not been considered in traditional DDoS attack defense. The enterprise must ensure its DDoS attack detection/defense operations neither affect nor be affected by other cloud users.

We illustrate these impacts using the example in Fig. 1. To ease the presentation, we denote an attacker in the private cloud of the enterprise network as a *local attacker*, an attacker in the off-site public cloud of the enterprise network as a *cloud attacker*, and other attackers as *outside attackers*. Similarly, we refer a server in the private cloud as a *local server* and a company's server in the public cloud as a *cloud server*. We consider two attacking scenarios. In the first attacking scenario, the victim server is within the private cloud. In the second one, the victim server resides in the public cloud.

In the first attack scenario, there are two types of attack traffic, i.e., (1) and (2) in Fig. 1. The enterprise's local DDoS attack defense system can detect the attack traffic (2), while the detection of the attack traffic (1) depends on whether the internal traffic is redirected to the DDoS attack defense system. Nevertheless, this scenario is similar to the traditional DDoS attack scenario. In what follows, we focus on two new challenges introduced in the second attacking scenario.

The first challenge is raised by the public accessibility of the cloud resources. We refer to this challenge as *extended defense perimeter*. There are three types of attack traffic, (3), (4) and (5). The enterprise's local defense can only examine and filter out the attack traffic (3) before the traffic leaves the local network. The defense offered by the cloud provider can check the attack traffic (4). However, more advanced attacks, such as the application-layer attacks which target specific applications, can bypass the generic defense provided by the cloud. The most stealthy attack is type (5) because it is initialized from the same physical network or even the same physical machine on which the application is running. Most of these traffic is handled at local switches or hypervisors without going through the detection hardware.

The second challenge is raised by the rapid resource reallocation. We refer to this challenge as *dynamic network topology*. This challenge makes the attack traffic (4) and (5) more difficult to handle because the enterprise's

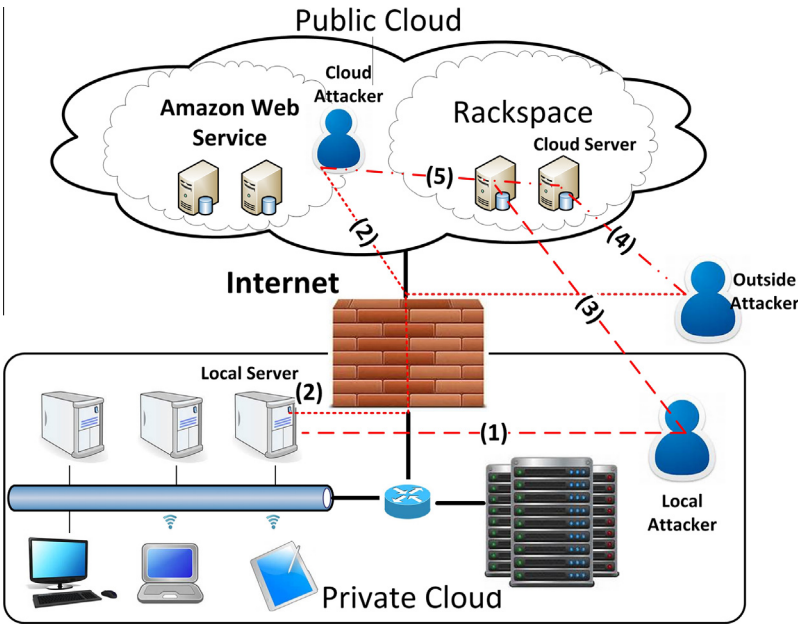


Fig. 1. The structure of a hybrid cloud, consisting of one private cloud and two public clouds. Five types of attack traffic are shown in the figure.

defense mechanism has to adjust to the network change caused by the physical location change of the virtual machine. The adaption must take effect in a short time period, for example, in milliseconds thanks to advances in live migration technology [11]. Moreover, because most of the topology changes are done by cloud provider without notifying the users, the DDoS attack defense mechanism needs to communicate with the cloud service provider to properly adapt the changes.

2.3. Software-Defined Networking

Unlike the well formatted data plane abstraction in the OSI model, the control plane of the Internet is composed of various complicated protocols for various network functions. Managing these protocols in a distributed manner becomes inefficient and error-prone. SDN is a network architecture that decouples the control plane and the data plane of network switches and moves the control plane to a centralized application called *network controller*. The network controller is in charge of the entire network through a vendor-independent interface such as OpenFlow [12], which defines the low-level packet forwarding behaviors in the data plane. Developers then can program the network from a higher level without concerning the lower level detail of packet processing and forwarding in physical devices.

2.4. Impact of SDN on DDoS attack defense

The most important two concepts of SDN are *control plane abstraction* and *network function virtualization*. They introduce following properties.

- *Centralized network control:* The centralized network operating system (NOS) connects to all the switches in the network directly. Thus, NOS can provide a global network topology along with the real-time network status.
- *Simplified packet forward:* The data plane in SDN simply forwards packets based on the forwarding policies generated by control programs.
- *Software based network function implementation:* Network functions originally implemented within a switch or a middle-box are implemented as control programs in SDN. These control programs reside above the NOS and communicate with switches remotely.
- *Virtualized network:* Similar to a hypervisor in hardware virtualization, the network virtualization hides the network topology from control programs so that network function developers can focus on the functionality implementation.

Implementing SDN affects the DDoS attack defense greatly in both directions. On the bright side, SDN makes advanced detection logic and rich subsequent processes easier to implement. On the downside, the devices or middle-boxes originally distributed within the network now need to be located above NOS. Compared with hardware-based packet processing, software processes packets much slower. The network delay and traffic overhead caused by the communications between the control program, i.e., the DDoS attack defense schemes, and the switches, may become the new attack surface.

2.5. DDoS attack defense in cloud computing and SDN

Based on our analysis, cloud computing introduces new DDoS challenges, i.e., *extended defense perimeter* and *dynamic network topology* due to its new operation model. To effectively address these challenges, the cloud provider must be able to (1) easily delegate the control of its network to cloud users; (2) fast re-configure the control according to the network topology changes caused by dynamic allocations and migrations. On one side, we could benefit from the centralized network controller and the network virtualization of SDN. On the other side, SDN influences DDoS attack defense in negative ways as we discussed early. The negative impact of SDN mainly comes from the efficiency of processing packets using software, which may generate new attack surface and lead to single-point failure. When designing a DDoS attack defense solution in SDN, one must take the computation and communication overhead into the consideration so that no new security vulnerability is introduced. To sum up, we believe SDN technology will benefit the DDoS attack defense in cloud computing as along as the design could carefully handle the communication and computation overhead.

3. DaMask design

3.1. Design overview

Based on the analysis in Section 2, we need to incorporate the DDoS attack defense into cloud computing and SDN. To successfully address the DDoS attack defense challenges in the new network environment, we must achieve the following objectives. First of all, the scheme must be *effective*. The design should be able to protect the services in both private and public clouds. It also should be able to adapt to the network topology changes and mitigate DDoS attacks efficiently. Secondly, the scheme should incur *small overhead*. The communication and computation overhead introduced by the architecture should also be limited to a small amount to be practical. Lastly, the deployment cost should be *inexpensive*. The solution should require as little deployment cost, such as additional hardware or changing existing protocols for both enterprises and cloud service providers, as possible.

To address the first challenge, our idea is to separate the enterprise's network traffic from the main network by virtualizing the network. We call such a virtual network a *slice*. Then we let the cloud provider delegate the slice to the owner of this slice. Similar with the hardware or platform virtualization, a slice contains the network flows related to the enterprise only and is isolated from other slices. The strong isolation between different slices ensures that a slice is visible to its belonging company only. Therefore, operations performed on the slice are transparent to other cloud users.

For the second issue, we should select an efficient attack detection algorithm which involves as little information as possible to reduce the communication overhead. Meanwhile, the detection process itself must be fast enough to incorporate with the packet forwarding speed. Existing DDoS attack detection algorithms could serve the purpose as long as it does not depend on certain hardware. It is also worth mentioning that signature-based detection or anomaly-based detection or even a combined detection scheme can be used here.

To cope with the last issue, we need a rapid re-configuration scheme for each slice in the cloud. Given the nature of a virtualized slice, which is defined by its profile, our idea is to re-configure each slice profile when the virtual machine migration is taking placing. Because the cloud provider virtualizes the network, he can track all the enterprises' controllers, and re-configure the profile of a slice when a migration is about to happen. By applying the new slice profile, the cloud provider ensures the right enterprise getting the control of the proper slice.

3.2. Workflow of DaMask

To substantiate our previous claim, we propose a DDoS attack mitigation architecture, named DaMask. The DaMask architecture has three layers, network switches, network controller, and network applications. The main functions of the DaMask are DDoS detection and reaction. There are two separate modules in the DaMask, *DaMask-D*, a network attack detection system, and *DaMask-M*, an attack reaction module. We present the workflow of DaMask in Fig. 2.

3.2.1. DaMask-D module

The DaMask-D module is an anomaly-based attack detection system. We argue that although signature-based attack detection could also work, they are not efficient. The reason is that, in SDN, the responsibility of generating a packet signature moves from a switch or a middle-box to a remote control program, which not only processes slower than hardware, but also requires all the packets to be redirect to it. Therefore, we focus on anomaly-based detection. Now we assume we already have a detection algorithm implemented (this can be done in an offline process as shown in Fig. 2).

In online phase, when a new packet arrives at the switch, the cloud provider first decides which slice the packet belongs to. Then the cloud provider notifies the corresponding NOS of the slice. After receiving the notification, the slice owner's NOS checks whether the packet belongs to an existing flow.[1] If so, it updates the flow statistic, otherwise it build a new flow record. Then we query the detection model with the updated or the new flow static. If the query result indicates an *attack*, DaMask-D issues an *alert* and forwards the alert along with the packet info to the DaMask-M module. If the query result is *normal*, the packet is forwarded to its intended destination. Occasionally, the detection model cannot determine the attack type of a packet if the packet belongs to a new type of attack. In that case, the packet needs to be further analyzed. The analysis result is then used to update the detection model through a *model updating* process.

[1] The flow definition varies for each slice according to different requirements of enterprise.

3.2.2. DaMask-M module

The DaMask-M module is an attack reaction system. In the existing work of DDoS protection in today's Internet, the reaction options are simple and limited, because advanced post-processing logic requires switches working together in a distributed manner. Implementing and managing such functions are time-consuming and error-prone. In SDN, we can implement those sophisticated logic such as quarantine of different types of packets to different location thanks to the control plane abstraction. The DaMask-M contains two functions: *countermeasure selection* and *log generation*. When DaMask-M receives an alert, it tries to match the alert to a countermeasure. The default action is to drop the packet if there is no pre-set policy for the alert. We implement DaMask-M as a set of common APIs so that defenders can customize their own defense countermeasure for different DDoS attacks. The basic unit a defender can play with is flow. We define three basic operations, *forward, drop* and *modify* to form advanced defense logic. Compared with DDoS attack mitigation in traditional network, DaMask-M provides a powerful way to implement the countermeasure. After the countermeasure is selected, DaMask-M pushes the policy to the switch through network controller. After that, the attack packet, along with its auxiliary information (e.g. the time stamp and response actions), is recorded in the log database.

4. Graphical model based detection system

In Section 3, we state that an anomaly-based network attack detection system will fit our DaMask framework well. In this section, we propose our attack detection system which is built on probabilistic inference graphical model. Although other existing attack detection systems are compilable with DaMask, our detection model advances with two features: (1) automatic feature selection; (2) efficient model update. By updating our model efficiently, we are able to address the *dataset shift* problem which is not considered in the existing schemes.

4.1. Graphical inference model

The core of the attack detection system is a graph model. It stores known traffic patterns as a relational graph between patterns and their labels (malicious or normal). When new network traffic arrives, the system uses this graph to determine whether it is malicious.

4.1.1. Automatic feature selection

To build the model for network traces, a set of features must be extracted from the network traces. In traditional anomaly-based detection systems, features are picked heuristically based on the designers' experience. Although experts can provide valuable insight, they may also

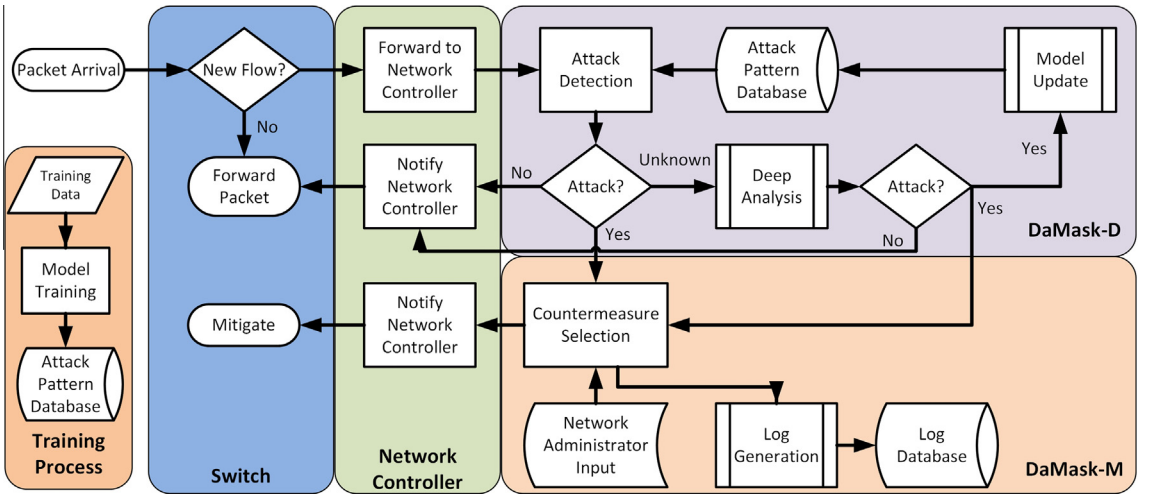


Fig. 2. Workflow of DaMask.

introduce inevitable bias due to their limited knowledge. A more objective way is to spawn a large candidate feature set and let the data decide which features are relevant. We exercise feature selection [13] on a large feature set. Considering the fact that network traffic is usually low dimension data (the number of cases is far greater than the number of features), the Chow–Liu algorithm [14] is a good choice because it surpasses other algorithms when learning from low dimension data [15].

Denote $\mathcal{X} = (X_1, X_2, \ldots, X_n)$ as the feature set, the Chow–Liu algorithm works as follows: (1) initialize an edgeless graph $G(V = \mathcal{X}, E = \emptyset)$ with each vertex corresponding to a feature; (2) for each pair of features $X_i, X_j \in \mathcal{X}$, perform an independency test using mutual information as the deviance measures:

$$I(X_i, X_j) = \sum_{a \in X_i} \sum_{b \in X_j} p(a,b) \log \left(\frac{p(a,b)}{p(a)p(b)} \right)$$

where a, b take all the possible values of features X_i, X_j respectively. The result is a weighted graph where the weight of an edge $e(X_i, X_j)$ is $I(X_i, X_j)$; (3) we compute a maximum spanning tree from the graph as the Chow–Liu tree. Since most relevant features are directly connected in a Chow–Liu tree, we exclude the redundant features from the model.

4.1.2. Attack detection

Upon receiving new network traffic, the system collects only those features selected by the Chow–Liu algorithm. In our design, all features we selected are observable, i.e., their values can be directly extracted from the packet content or flow statistic. Let the set of features $\mathcal{E}$ observed from a network flow is a subset of $\mathcal{X}$, Y is the class label of that flow, and $\mathcal{W} = \mathcal{X} - Y - \mathcal{E}$ be those features that are not observable, such as encrypted payload. The attack detection process is a maximum a posterior (MAP) query, i.e., finding the optimal assignment to all of the (non-evidence) features $Y \cup \mathcal{W}$ given the evidence $\mathcal{E} = e$:

$$\text{MAP}(Y, \mathcal{W} | e) = \arg\max_{y,w} P(y, w | e)$$

When all features are observed, i.e., $\mathcal{W} = \emptyset$, the MAP query further reduces to a conditional probability query:

$$P(Y|e) = \frac{P(Y, e)}{P(e)}$$

4.2. Graph model update

Most of the existing works assume that the actual attack patterns follow the same true distribution as in training dataset. Sadly, it is not true. In reality, the traffic pattern is influenced a lot by temporal and spatial factors [16]. The problem is known as dataset shift problem [17].

To cope with the dataset shift in network traffic data, the system should tune the graph model based on new observed data. We consider two types of update depending on the deviance between new attack patterns and existing ones. If the deviance is large, a global update is required, which searches a new graph structure. However, the global update is too costly to be performed frequently. Therefore, when the deviance is small, we perform a local update, which updates the conditional probability of the nodes in the graph model.

The idea of the local update is to estimate the $P(Y)$, i.e., the distribution of the traffic types (normal or malicious), based on the newly observed data. Then we can use the new $P(Y)$ to update the conditional probability distribution (CPD) of the features used in the attack detection. The local update process is efficient because it does not involve the graph structure change. In our scheme, the variable Y indicating the traffic type follows a *multinomial distribution* with k parameters $\boldsymbol{\theta} = (p_1, p_2, \ldots, p_k)$, $k \in \mathbb{Z}^+$. We use a point estimator to estimate the $\boldsymbol{\theta}$ using the newly observed data. The process works as follows. First, we model $P(\boldsymbol{\theta})$ using a Dirichlet distribution with parameters $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ as

$$Dir(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \frac{\Gamma(\boldsymbol{\alpha}k)}{\Gamma(\boldsymbol{\alpha})^k}\prod_1^k p_i^{\boldsymbol{\alpha}-1}.$$

Then we use the following equation to estimate the parameters $\widehat{\boldsymbol{\theta}}$ for $P_{test}(Y)$, i.e., the new distribution of variable Y.

$$P(\boldsymbol{\theta}|E) = \frac{P(E|\boldsymbol{\theta})P(\boldsymbol{\theta})}{\sum_{\boldsymbol{\theta}} P(E|\boldsymbol{\theta})P(\boldsymbol{\theta})},$$

where E is the newly observed data.

$$P(\boldsymbol{\theta}|E) \propto Dir(\boldsymbol{\theta}|\widehat{\boldsymbol{\alpha}}),$$

where $\widehat{\boldsymbol{\alpha}} = (\alpha_1 + N_1, \alpha_2 + N_2, \ldots, \alpha_k + N_k), N_i$ is the data count in the new observation, of which Y values are equal to y_i. At last, the parameters $\widehat{\boldsymbol{\theta}}$ for $P_{test}(Y)$ can be easily estimated as

$$\widehat{\theta}_i = \frac{N_k}{N}, \qquad \forall i, \quad i \leqslant k, \quad i \in \mathbb{Z}^+.$$

This equation indicates that we can update our graph model by only updating the local conditional probability of each variable connected with the attack type Y in the graph model, which is computationally inexpensive.

Theoretically, updating the local CPD is enough if the underlying graph structure captures the relationship among the variables precisely. However, the precision of the graph structure is hard to measure in reality. If the time interval between the local update and the last global update is relatively short, the estimation result is good enough to approximate the joint distribution. We further study the impact and present the results in Section 5.

5. DaMask evaluation

We carried out a thorough performance evaluation of the DaMask architecture under various scenarios. We run detection accuracy test on our attack detection system using real world network traffic The evaluation results are reported in this section.

5.1. Evaluation setting

To evaluate the performance of the DaMask, we have set up a hybrid cloud. We use Amazon Web Service EC2 as our public cloud while we simulate the private cloud in our lab. The overall evaluation environment is shown in Fig. 3. We utilize Mininet [18], which creates a realistic virtual network on a computer, to emulate the SDN setting.

5.1.1. Private cloud

The private cloud consists of two Linux servers in our lab. Both of them are running Ubuntu 12.10 32-bit operating system. One laptop (denoted as Linux A), which equips with AMD E1-1200 × 2 at 1.4 GHz CPU and 4 GB memory, emulates the private cloud. The other desktop (denoted as Linux B) equips with Intel i7-2600 CPU at 3.4 GHz and 12 GB memory runs the network controller and DaMask with our attack detection system on it. Linux A and Linux B are connected through a Intel Express 460T 100 MB switch. We choose Floodlight [19] as the network controller

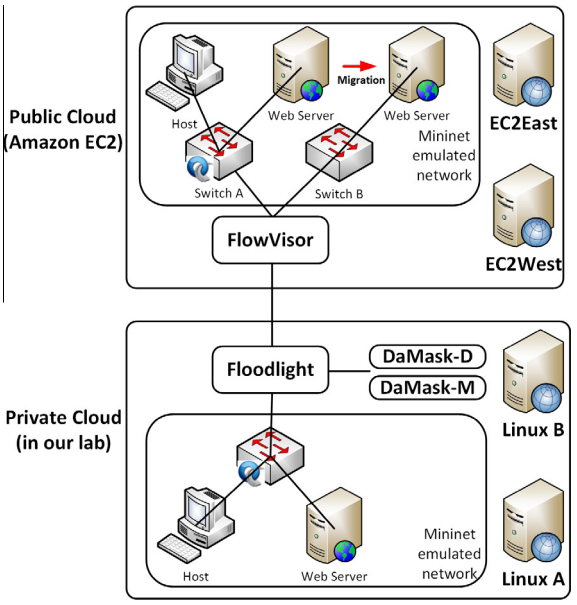


Fig. 3. The simulated hybrid cloud topology.

since Floodlight controller can be easily extended and enhanced through its module loading system.

We emulate a virtual network using Mininet in Linux A to extend the private cloud. The private cloud in Fig. 3 has one switch and two hosts. One of the hosts is an web server (Apache Http server 2.2.26). The Floodlight controller and the DaMask modules are deployed in Linux B. The DaMask modules communicate with the controller through Floodlight's APIs.

5.1.2. Public cloud

To measure the communication cost of DaMask, we use the Amazon Web Service (AWS) EC2 as our public cloud in our evaluation. We deployed two AWS EC2 instances as the company's remote web servers. Both of them are Ubuntu T1-Micro instances. One of them (denoted as EC2West) is located at US West (Oregon) and the other (denoted as EC2East) is located at US East (N. Virginia).

We use EC2West, which runs FlowVisor to handle network virtualization, to simulate the network administration of the public cloud. We emulate a virtual network in EC2East to extend the remote side of the company's network, which is the public cloud part in Fig. 3. Similar with the private cloud extension, the extended public cloud also has one switch and two hosts, one of which is an Apache web server. The difference is that the switch is connected to the FlowVisor in EC2West instead of a network controller.

5.1.3. Evaluation dataset

To evaluate the attack detection performance of our graphical model based detection module, we adopt the UNB ISCX dataset [20]. The UNB ISCX dataset labeled the DDoS attack network traffic, which means we have ground truth of the traffic. We extracted 18 features from the network traces. We divide the entire data set into ten equal

shares. The first partition and the last partition are used as the training data and the testing data, respectively, while the other eight parts are used for the online model update process.

5.2. DaMask overhead

5.2.1. Computation cost of attack detection

The computation overhead comes from three aspects: (1) the offline graphical model training process; (2) the online testing process; and (3) the model updating process. We implemented DaMask-D module using R language and trained the model with the UNB ISCX dataset. Fig. 4 shows the computation costs of the model building process.

As we mentioned before, the network traffic pattern difference between the training phase and the testing phase leads to inaccurate detection result in practice. We evaluate the overhead of local update as follow. First, we use 10% of the data as the training data, 10% of the data as the testing data and divide the remain data to 8 update datasets. Then we use the training dataset to generate a model, denoted as $\mathcal{M}_{basic}$. After that, we perform a global update and an iterative local update using the updating datasets to get two new models, denoted as $\mathcal{M}_{global}$ and $\mathcal{M}_{local}$ respectively. The computation time of both processes is shown in Fig. 4.

From the figure, we can see the model generation time is a linear function with respect to the number of data in the dataset; while the local update time is only related to the number of the data in the new observation. The simulation result validates the claim we have made in Section 4.2, i.e., the cost of a local update is much cheaper than the cost of a global update. We delay the detection accuracy comparison between global update and local update in Section 5.4.

5.2.2. Communication overhead

DaMask introduces communication overhead since now the traffic towards the servers in the public cloud needs to be examined by the DaMask-D module located at the enterprise's local network. To evaluate the communication overhead, we carried out several experiments.

We first measured the network bandwidth of our evaluation environment. We measure the network bandwidth by running iperf 2 times an hour for a consecutive 24 h. The average bandwidth between Linux A and EC2West is 27.4 MB/s, and the average bandwidth between Linux A and EC2East is 86.2 MB/s. The connection between Linux A and EC2East is better because our lab is located at east coast. We then tested the response time from the remote server with and without DaMask being deployed. We show our result in Table 1. The results show that the communication overhead is only related to the round trip time between the server running the FlowVisor in the public cloud and the server running the network controller in the private cloud. This is because we fixed the size of the message to be sent to the network controller. Therefore, the communication overhead is a constant if the link status of network is stable.

5.3. Adapting topology change

One advantage of DaMask is that DaMask is able to adapt the network topology change caused by virtual machine migrations. To simulate the migration process, we added an additional switch, i.e., switch B in Fig. 3. Suppose the web server is migrated from the switch A to

Table 1
Communication time.

Task	Basic		DaMask w/o test		DaMask w/test	
	West	East	West	East	West	East
Ping (ms)	196	12	425	51	462	85
Http (s)	2.4	1.7	2.3	1.6	2.4	1.6

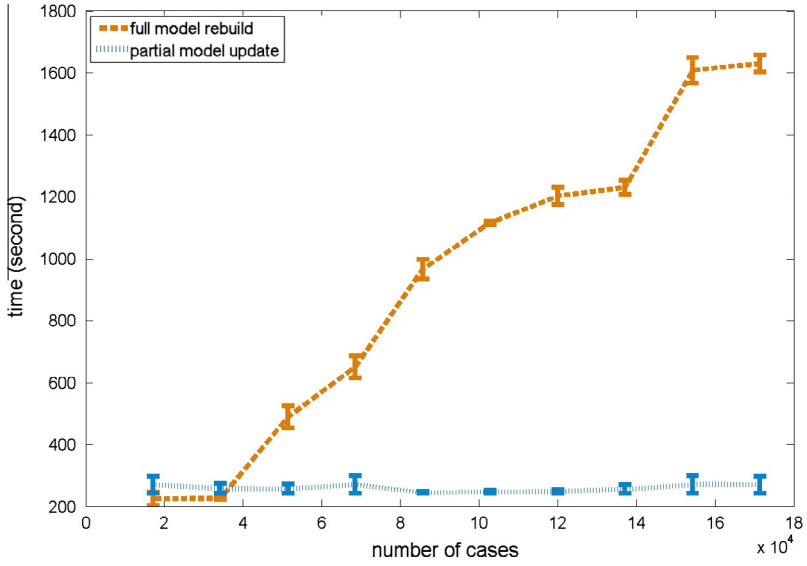


Fig. 4. Running time vs. # of the training data.

the switch B, DaMask need taking control of the switch B while dropping control of the switch A since the switch A no longer belongs to the company's slice. Such re-configuration is accomplished by changing the flow space header of the company's slice in FlowVisor. Re-configuration in FlowVisor can be efficiently done through Command Line Interface (CLI) of FlowVisor. Since FlowVisor can reload the new slice configuration without interrupting the service, this process can be done in real-time.

After changing the flow space of the company's slice, we sent ICMP packets to the web server that is attached to the switch B. All the ICMP packets were received by the company's controller, which verifies the company indeed had the control of the switch B. We further tested if the company's control slice affected other users. We sent ICMP packets to the web server linked with the switch A and none of the packet was received by the company's controller, which means the FlowVisor did not forward any ICMP packet to the company's network controller.

5.4. Detection performance

5.4.1. Data shift

We first use the data set to demonstrate that there exists data shift issue in network traffic. In practice, the detection model is built with a training set which is always a per-collected traffic data while the detection is performed over new traffic data. In our simulation, we use part of the data as training data to build the model, i.e., $\mathcal{M}_{basic}$. We also build a model, i.e., $\mathcal{M}_{global}$. As shown in Fig. 5, the model is different. Indeed, the attack traffic features different characteristics during different period as mentioned in [20]. Therefore, it is necessary to update the detection model in real-time to ensure the detection performance.

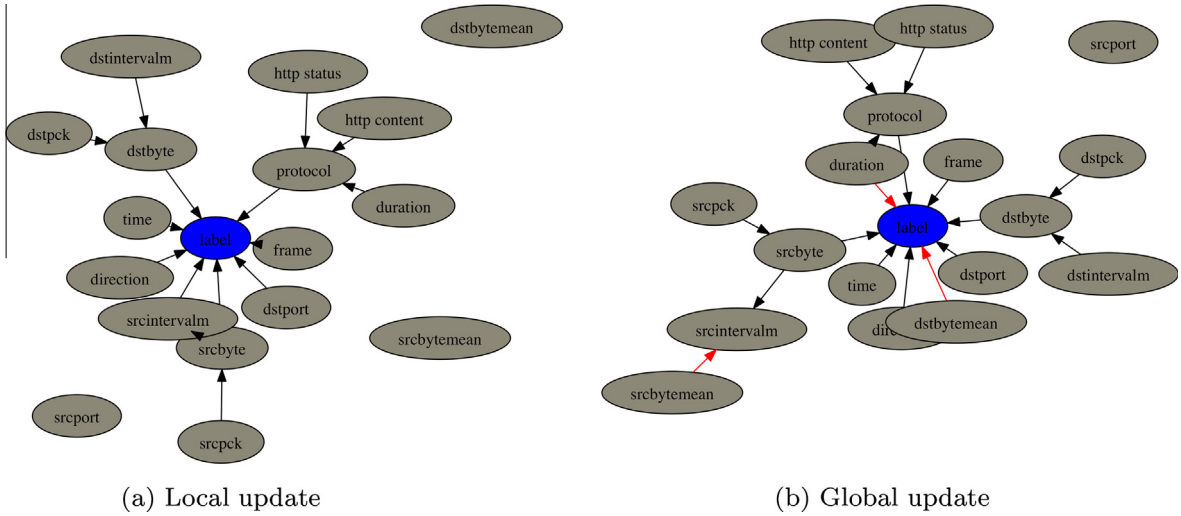
Although performing a global update, i.e., building a new model based on the new observation data, solves the data shift issue, it is always too expensive to adopt in reality. Our approach which is update the CPDs on the original model mitigates the impact of the issues and therefore, improves the detection accuracy as demonstrated in next subsection.

5.4.2. Accuracy

The last evaluation is to test the detection accuracy of our attack detection system. Table 2 shows the detection rates for all three our models, $\mathcal{M}_{basic}$, $\mathcal{M}_{global}$ and $\mathcal{M}_{local}$. The *detection rate* is the number of detected attacks divided by the number of total attacks. The *miss detection rate* is the number of missed attacks divided by the number of total attacks. We also reported the false positive rate and the true positive rate in the form of ROC curve in Fig. 6. As we expected, there is a performance degradation between training and testing set due to the data shift. Such degradation can be remedied by both local or global updates. $\mathcal{M}_{local}$ performs better than $\mathcal{M}_{basic}$ but a little worse than $\mathcal{M}_{global}$. This is because local update can only remedy data shift deviation which is not big to change the model structure. Over time, the data shift deviation will cause the structure change of the underlying graphical model. At that time, a global update is needed. We show the graph structure learned from the training data in Fig. 5. We can see the feature selection property of our model from the figure, i.e., not all features are related to attack detection. And the graph model structure trained through local update (Fig. 5a) is different from the one trained through global update (Fig. 5b). It is worth mentioning that even the underlying graph structure changes, the detection accuracy will not be degraded greatly.

Table 2
Detection accuracy.

	Detection rate (%)	Miss detection rate (%)
$\mathcal{M}_{basic}$	74.02	25.98
$\mathcal{M}_{local}$	86.56	13.44
$\mathcal{M}_{global}$	89.30	10.70



(a) Local update (b) Global update

Fig. 5. Graph structures.

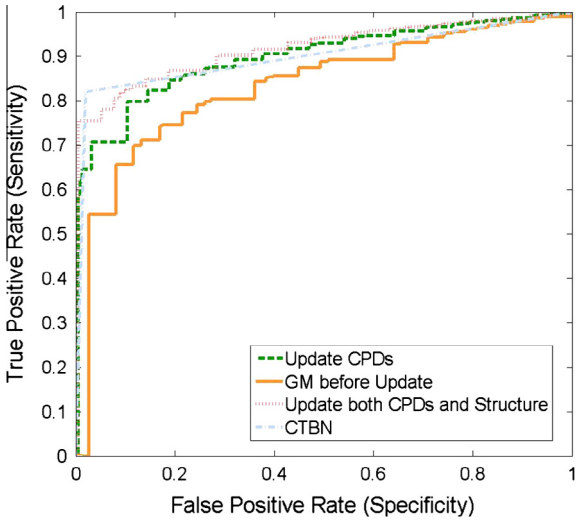


Fig. 6. ROC curves for $\mathcal{M}_{basic}$ (GM), $\mathcal{M}_{global}$ (update both CPDs and structure) and $\mathcal{M}_{local}$ (update CPDs only) and CTBN.

5.4.3. Comparison

We first compare our detection scheme with the scheme in [9] which used continuous time Bayesian network (CTBN). The ROC curve in Fig. 6 shows that the performance of our method is similar to theirs. However, our model excels in terms of smaller computational cost because latent variables are included in CTBN. In order to learn such variables, they used the EM algorithm which requires performing an inference for each iteration. Compared to that, our design does not contain any latent variables, and therefore does not need to perform inference during learning. Also, our model enjoys higher expressiveness compare to their CTBN template since we imposed less structural constraints.

We also compare our detection results with Snort and Snort.AD. Snort is a popular open-source, signature-based network intrusion detection while Snort.AD is an anomaly-based preprocessor for Snort, which uses Holt–Winters model to detect anomaly network behavior including DDoS attacks. Snort reported 6.73% attack packets because most of the attack packets are well-formatted HTTP requests, which can bypass the predefined signatures. Snort.AD, on the other hand, generated 23 more alerts than Snort, but only two of them are real attack. The reason that Snort.AD works poorly is due to the stealth of the application layer DDoS attack.

6. Related work

Defending DDoS attack in traditional network has been studied for several decades. The surveys [21,22] have included most of these work. Although our objective shares the similarity with them which is to defend DDoS attacks, our network environment which involves cloud computing and SDN is quite different from theirs. SDN technique has been used to address various network security. Jafarian et al. [23] proposed a random host mutation scheme using OpenFlow to achieve transparent moving target defense in SDN. Porras et al. proposed a security enforcement kernel for SDN in [24] to detect policy conflicts within the switches. Yao et al. utilized the SDN architecture to validate source addresses in [25]. The key difference between those work and ours is that they try to address the traditional network security threats using SDN to achieve better performance while we focus on the new challenges in the new network paradigm. Recently, Shin et al. [26] proposed an OpenFlow security application development framework, FRESCO, to enhance the secure application development in SDN. In contrast with FRESCO, our work focuses on DDoS attack challenges in cloud computing, which requires additional functionalities such as letting enterprises control the network slice other than those provided by existing solutions.

A network intrusion detection system (NIDS) differentiates malicious traffic from the benign traffic. There are two broad categories of IDSes: signature-based IDS and anomaly-based IDS. Signature-based detection systems, e.g. Snort, can detect known attacks by utilizing the signature of those attacks. Such systems require frequent signature updates and could only detect known attacks. Anomaly-based detection systems are able to detect abnormal network traffics which could potentially be attacks. Patcha and Park presented a survey of existing anomaly detection techniques in [27]. One of the popular techniques in anomaly detection is Bayesian network inference model, which has several advantages for data analysis [28]. Kruegel et al. [7] proposed a Bayesian classification algorithm to do intrusion detection by monitoring the system calls. Gupta et al. [8] incorporated multiple detection layers, all of which are Bayesian network based, to increase the detection accuracy. Xu and Shelton [9] used a continuous time Bayesian network model, which considers temporal sequence of events, to construct both network-based and host-based intrusion detection systems. Although we use the Bayesian network inference model to detect the DDoS attacks as well, the major differences between those works and ours is that our graphical model updates itself based on new observations continuously to address the potential dataset shift issue.

7. Conclusion

Cloud computing is already here to stay and SDN is gaining increased popularity. With both of the technology emerging as the future enterprise IT solutions, it is worthwhile to look at the implications of the combination of the two, particularly on the enterprise network security. In this paper, we analyze the impact of cloud computing and SDN on DDoS attack defense. Based on our analysis, we identify the challenges and the benefits raised by these new technologies. We claim that with careful design, SDN could help with DDoS attack protection. To substantiate our finding, we proposed our solution of defending DDoS attack—DaMask architecture. Compared to the existing solutions, DaMask requires little effort from the cloud provider which means few changes are required from the current cloud computing service architecture. The SDN-based network monitoring and control mechanism allow companies to control and configure their defense mechanisms in

the cloud effectively without affecting other cloud users. In addition, DaMask features a graphical model based anomaly detection module. To enhance the detection accuracy, we proposed a model update method that updates the inference model periodically using Bayesian inference method. We also carried out a simulation study using real network traces to evaluate the performance. The results show that our proposed DaMask is successful in dealing with the new challenges raised. The SDN-based network management can rapidly adapt to the network topological changes. The detection algorithm is fast enough to perform online packet inference and it achieves a high detection rate. The proposed model update process saves a significant amount of time compared to regenerating a model while suffering hardly any performance loss in terms of detection accuracy.

Acknowledgment

We gratefully acknowledge funding support for this research from U.S. National Science Foundation under Grant CNS-1217889.

References

[1] D. Geneiatakis, G. Portokalidis, A.D. Keromytis, A multilayer overlay network architecture for enhancing IP services availability against DoS, in: 7th International Conference,ICISS, Kolkata, India, December 2011.
[2] X. Liu, X. Yang, Y. Lu, To filter or to authorize: network-layer DoS defense against multimillion-node botnets, in: ACM SIGCOMM Computer Communication Review, ACM, 2008.
[3] P. Mittal, D. Kim, Y.C. Hu, M. Caesar, Mirage: Towards Deployable DDoS Defense for Web Applications, arXiv preprint, August 2012.
[4] W.G. Morein, A. Stavrou, D.L. Cook, A.D. Keromytis, V. Misra, D. Rubenstein, Using graphic turing tests to counter automated DDoS attacks against web servers, in: 10th ACM Conference on CCS, ACM, 2003.
[5] H. Wang, L. Xu, G. Gu, Of-guard: A DoS Attack Prevention Extension in Software-defined Networks, The Open Network Summit (ONS), 2014.
[6] D. Kreutz, F. Ramos, P. Verissimo, Towards secure and dependable software-defined networks, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software defined Networking, ACM, 2013, pp. 55–60.
[7] C. Kruegel, D. Mutz, W. Robertson, F. Valeur, Bayesian event classification for intrusion detection, in: Computer Security Applications Conference, 2003, Proceedings, 19th Annual, IEEE, 2003.
[8] K.K. Gupta, B. Nath, R. Kotagiri, Layered approach using conditional random fields for intrusion detection, IEEE Trans. Dependable Sec. Comput. (2010).
[9] J. Xu, C.R. Shelton, Intrusion detection using continuous time bayesian networks, J. Artif. Intell. Res. (2010).
[10] B. Wang, Y. Zheng, W. Lou, Y.T. Hou, DDoS attack protection in the era of cloud computing and software-defined networking, in: 9th Workshop on Secure Network Protocols (NPSec) in Conjunction with ICNP, IEEE, 2014.
[11] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, A. Warfield, Live migration of virtual machines, in: Proceedings of the 2nd Conference on Symposium NSDI, USENIX, 2005.
[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, Openflow: enabling innovation in campus networks, ACM SIGCOMM Comput. Commun. Rev. (2008).
[13] D. Kollar, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, The MIT Press, 2009.
[14] C. Chow, C. Liu, Approximating discrete probability distributions with dependence trees, IEEE Trans. Inform. Theory (1968).
[15] K.P. Murphy, Machine Learning: A Probabilistic Perspective, The MIT Press, 2012.
[16] K. Thompson, G. Miller, R. Wilder, Wide-area internet traffic patterns and characteristics, IEEE Netw. (1997).
[17] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, N.D. Lawrence, Dataset Shift in Machine Learning, The MIT Press, 2009.
[18] B. Lantz, B. Heller, N. McKeown, A network in a laptop: rapid prototyping for software-defined networks, in: 9th ACM SIGCOMM Workshop HotSDN, ACM, 2010.
[19] Floodlight OpenFlow Controller. <http://floodlight.openflowhub.org>.
[20] A. Shiravi, H. Shiravi, M. Tavallaee, A.A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, 2012.
[21] T. Peng, C. Leckie, K. Ramamohanarao, Survey of network-based defense mechanisms countering the dos and DDoS problems, ACM Comput. Surv. (1) (2007).
[22] S. Zargar, J. Joshi, D. Tipper, A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks, IEEE Commun. Surv. Tutorials (2013).
[23] J.H. Jafarian, E. Al-Shaer, Q. Duan, Openflow random host mutation: transparent moving target defense using software defined networking, in: Proceedings of the 1st Workshop on HotSDN, SIGCOMM, ACM, 2012.
[24] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, G. Gu, A security enforcement kernel for openflow networks, in: Proceedings of the 1st Workshop on HotSDN, SIGGCOMM, ACM, 2012.
[25] G. Yao, J. Bi, P. Xiao, Source address validation solution with openflow/noxarchitecture, in: IEEE ICNP, 2011.
[26] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, M. Tyson, Fresco: modular composable security services for software-defined networks, in: Proceedings of NDSS, 2013.
[27] A. Patcha, J.M. Park, An overview of anomaly detection techniques: existing solutions and latest technological trends, Comput. Netw. (2007).
[28] D. Heckerman, A Tutorial on Learning with Bayesian Networks, Springer, 2008.

Bing Wang received his BS and ME degree in Computer Science from Fudan University and Shanghai Jiao-Tong University, respectively. He is currently working towards Ph.D. degree in Computer Science at Virginia Tech. His research interests are in the areas of applied cryptography and network security, with current focus on secure data service in cloud computing and next generation Internet. He is a student member of the IEEE.

Yao Zheng received the BS degree in microelectronic from Fudan University and the MS degree in electrical engineering from Worcester Polytechnic Institute. He is currently working toward the PhD student at Virginia Tech. His current interest are in android application security and linux kernel development. He is a student member of the IEEE.

Wenjing Lou is a Professor at Virginia Polytechnic Institute and State University. Prior to joining Virginia Tech in 2011, she was a faculty member at Worcester Poly-technic Institute from 2003 to 2011. She received her Ph.D. in Electrical and Computer Engineering at the University of Florida in 2003. Her current research interests are in cyber security, with emphases on wireless network security and data security and privacy in cloud computing. She was a recipient of the U.S. National Science Foundation CAREER award in 2008.

Y. Thomas Hou is a Professor in the Bradley Department of Electrical and Computer Engineering, Virginia Tech. His research interest-s are cross-layer optimization for wireless networks. He is also interested in wireless security. Professor Hou is currently serving as an Area Editor of IEEE Transactions on Wireless Communications, an Associate Editor of IEEE Transactions on Mobile Computing, an Editor of IEEE Journal on Selected Areas in Communications, and an Editor of IEEE Wireless Communications. He is the Chair of IEEE INFOCOM Steering Committee.