ISSN:0975-9646

Employing a Reconfigurable Virtual Networking Approach by using NICE Mechanism

A.JensilinMary.M.E , E.SathishKumar , S.NavinChander , R.Naveenkumar , A.Prasanth

Department of Information Technology
SKP Engineering College, Tiruvannamalai-606611, TamilNadu

Abstract-Today's world moving around cloud computing technology ,the cloud computing plays a major role in all organizations this is because of its property like computability, cost efficiency ,availability etc.. The cloud computing had more advantage however it also have some security defect, In the cloud server detection of zombie exploration attacks is extremely difficult, due to this the cloud user can able to install harmful applications into their virtual server to attack the virtual server.

In this paper we provide Reconfigurable virtual network approach by using Network Intrusion Detection and Countermeasure selection algorithm ,this algorithm prevent harmful attack of virtual network system by the user .In this method, using attack analyzer we analyze the vulnerability of the application which is uploaded and downloaded by the cloud user if the application is harmful to the virtual network system the analyzer will restrict the application ,it works like bridge between the distributed virtual network system in order to significantly improve attack detection and mitigate attack consequences. The data efficiency and the security of the cloud computing is improved effectively.

I.INTRODUCTION

In Recent studies have shown that users migrating to the cloud consider security as the most important factor. A recent Cloud Security Alliance (CSA) survey shows using of cloud computing is considered as the top security threat, in which attackers can exploit vulnerabilities in clouds and utilize cloud system resources to attack it. In data centers, the system administrators have full control over the host system, defect can be detected and rectified by the system administrator. However, rectifying known security holes in cloud server, where cloud users usually have the rights to control software installed on their managed VMs, may not work properly and can affects the *Service Level Agreement* (SLA). Furthermore, cloud users can install vulnerable software on their VMs, which leads to loopholes in security. The difficult is to establish an effective vulnerability/attack detection and response system for accurately identifying attacks and minimizing the impact of security breach to cloud users. In, M. Armbrust *et al.* addressed that protecting"Business continuity and services availability" from service outages is one of the top concerns in cloudComputing systems.

In a cloud system where the infra structure is shared by potentially many users attacked by use of the shared infrastructure benefits attackers to exploit vulnerabilities of the cloud and use its resource to deploy attacks in more efficient ways . Such attacks are more effective in the cloud environment since cloud users usually share computing resources, e.g., sharing the same data and file systems, even with attackers. The same setup for VMs in the cloud, e.g., virtualization method, VM OS, installed harmful software, networking, etc., attracts attackers to attack multiple VMs. In this article, we propose NICE (Network Intrusion detection and Counter measures Election in virtual network systems) to establish a defense-in-depth intrusion detection. For attack detection, NICE uses attack graph analytical procedures into the intrusion detection processes. NICE employs a reconfigurable virtual networking approach to detect and counter the attempts to attack VMs, that preventing zombie VMs.

In general, NICE includes two main phases: (1) deploya lightweight mirroring-based network intrusion detection agent (NICE-A) on each cloud server to capture and analyze cloud traffic. A NICE-A repeatedly scans the virtual system vulnerabilities within a cloud server to establish Scenario Attack Graph (SAGs), and then based on the severity of identified vulnerability. (2) Once a VM enters inspection state, (DPI) Deep Packet Inspection is applied, virtual network reconfigurations can be deployed to the inspecting VM to make the potential attack behaviors prominent.

II.PROPOSED SYSTEM

In this article, we propose NICE (Network Intrusion detection and Countermeasure selection in virtual network systems) to establish a defense-in-depth intrusion detection framework. For better attack detection, NICE incorporates attack graph analytical procedures into the intrusion detection processes. We must note that the design of NICE does not intend to improve any of the existing intrusion detection algorithms; indeed, NICE employs a reconfigurable virtual networking approach to detect and counter the attempts to compromise VMs, thus preventing zombie VMs.

Advantage of proposed system

- The contributions of NICE are presented as follows:
- We devise NICE, a new multi-phase distributed network intrusion detection and prevention framework in a virtual networking environment that captures and inspects suspicious cloud traffic without interrupting users' applications and cloud services.
- NICE incorporates a software switching solution to quarantine and inspect suspicious VMs for further investigation and protection. Through programmable network approaches, NICE can improve the attack detection probability and improve the resiliency to VM exploitation attack without interrupting existing normal cloud services.
- NICE employs a novel attack graph approach for attack detection and prevention by correlating attack behavior and also suggests effective countermeasures.

NICE optimizes the implementation on cloud servers to minimize resource consumption. Our study shows that NICE consumes less computational overhead compared to proxy-based network intrusion detection solutions.

III.EXISTING SYSTEM

Cloud users can install vulnerable software on their VMs, which essentially contributes to loopholes in cloud security. The challenge is to establish an effective vulnerability/attack detection and response system for accurately identifying attacks and minimizing the impact of security breach to cloud users. In a cloud system where the infrastructure is shared by potentially millions of users, abuse and nefarious use of the shared infrastructure benefits attackers to exploit vulnerabilities of the cloud and use its resource to deploy attacks in more efficient ways. Such attacks are more effective in the cloud environment since cloud users usually share computing resources, e.g., being connected through the same switch, sharing with the same data storage and file systems, even with potential attackers. The similar setup for VMs in the cloud, e.g., virtualization techniques, VM OS, installed vulnerable software, networking, etc., attracts attackers to compromise multiple VMs.

Disadvantage of exiting system:

- No detection and prevention framework in a virtual networking environment.
- Not accuracy in the attack detection from attackers.

IV.System Modules

Nice-A

The NICE-A is a Network-based Intrusion Detection System (NIDS) agent installed in each cloud server. It scans the traffic going through the bridges that control all the traffic among VMs and in/out from the physical cloud servers. It will sniff a mirroring port on each virtual bridge in the Open vSwitch. Each bridge forms an isolated subnet in the virtual network and connects to all related VMs. The traffic generated from the VMs on the mirrored software bridge will be mirrored to a specific port on a specific bridge using SPAN, RSPAN, or ERSPAN methods. It's more efficient to scan the traffic in cloud server since all traffic in the cloud server needs go through it; however our design is independent to the installed VM. The false alarm rate could be reduced through our architecture design.

VM Profiling

Virtual machines in the cloud can be profiled to get precise information about their state, services running, open ports, etc. One major factor that counts towards a VM profile is its connectivity with other VMs. Also required is the knowledge of services running on a VM so as to verify the authenticity of alerts pertaining to that VM. An attacker can use port scanning program to perform an intense examination of the network to look for open ports on any VM. So information about any open ports on a VM and the history of opened ports plays a significant role in determining how vulnerable the VM is. All these factors combined will form the VM profile. VM profiles are maintained in a database and contain comprehensive information about vulnerabilities, alert and traffic.

Attack Analyzer

The major functions of NICE system are performed by attack analyzer, which includes procedures such as attack graph construction and update, alert correlation and countermeasure selection. The process of constructing and utilizing the Scenario Attack Graph (*SAG*) consists of three phases: information gathering, attack graph construction, and potential exploit path analysis. With this information, attack paths can be modeled using SAG. The Attack Analyzer also handles alert correlation and analysis operations. This component has two major functions: (1) constructs Alert Correlation Graph (*ACG*), (2) provides threat information and appropriate countermeasures to network controller for virtual network reconfiguration. NICE attack graph is constructed based on the following information: *Cloud system information, Virtual network topology and configuration information, Vulnerability information*

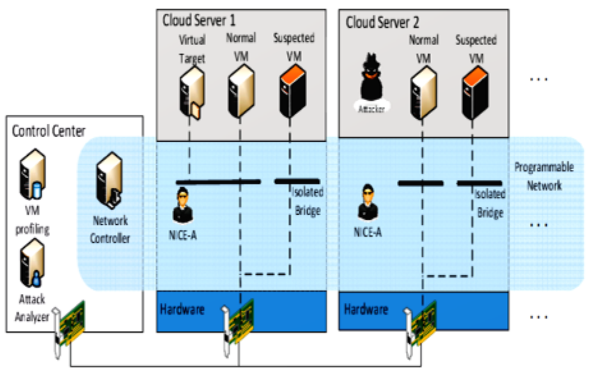
Network Controller

The network controller is a key component to support the programmable networking capability to realize the virtual network reconfiguration. In NICE, we integrated the control functions for both OVS and OFS into the network controller that allows the cloud system to set security/filtering rules in an integrated and comprehensive manner. The network controller is responsible for collecting network information of current Open Flow network and provides input to the attack analyzer to construct attack graphs. In NICE, the network control also consults with the attack analyzer for the flow access control by setting up the filtering rules on the corresponding OVS and OFS. Network controller is also responsible for applying the countermeasure from attack analyzer. Based on *VM Security Index* and severity of an alert, countermeasures are selected by NICE and executed by the network controller.

Algorithm 1 Alert Correlation

```
Require: alert ac, SAG, ACG
1: if (ac is a new alert) then
2: create node ac in ACG
3: n1 ← vc∈map(ac)
4: for all n2 ∈parent(n1) do
5: create edge (n2.alert, ac)
6: for all Si containing a do
7: if a is the last element in Si then
8: append ac to Si
9: else
10: create path Si+1 = {subset (Si, a), ac}
11: end if
12: end for
13: add ac to n1.alert
14: end for
15: end if
16: return S
```

V.System Design



NICE architecture within one cloud server cluster.

In this section, we first present the system design overview of NICE and then detailed descriptions of its component The proposed NICE framework is illustrated in figure 1.It shows the NICE framework within one cloud server cluster. Major components in this framework are distributed and light-weighted NICE-A on each physical cloud server, a network controller, a VM profiling server, and an attack analyzer. The latter three components are located in a centralized control center connected to software switches on each cloud server (i.e., virtual switches built on one or multiple Linux software bridges). NICE is a software agent implemented in each cloud server connected to the control center through a dedicated and isolated secure channel, which is separated from the normal data packets using Open Flow tunneling or VLAN approaches. The network controller is responsible for deploying attack countermeasures based on decisions made by the attack analyzer.

In the following description, our terminologies are based on the XEN virtualization technology. NICE-A is a network intrusion detection engine that can be installed in either Dom0 or Dom U of a XEN cloud server to capture and filter malicious traffic. Intrusion detection alerts are sent to control center when suspicious or anomalous traffic is detected. After receiving an alert ,attack analyzer evaluates the severity of the alert based on the attack graph, decides what countermeasure strategies to take, and then initiates it through the network controller. An attack graph is established according to the vulnerability information derived from both offline and real time vulnerability scans. Offline scanning can be done by running penetration tests and online real time vulnerability scanning can be triggered by the network controller (e.g., when new ports are opened and identified by Open Flow switches) or when new alerts are generated by the NICE-A. Once new vulnerabilities are discovered or countermeasures are deployed, the attack graph will be reconstructed. Countermeasures are initiated by the attack analyzer based on the evaluation results from the cost-benefit analysis of the effectiveness of countermeasures. Then, the network controller initiates counter measure actions by reconfiguring virtual or physical Open Flow switches.

VI.NICE SECURITY MEASUREMENT, ATTACK MITIGATION AND COUNTERMEASURES

In this section, we present the methods for selectingthe countermeasures for a given attack scenario. When vulnerabilities are discovered or some VMs are identifiedas suspicious, several countermeasures can be taken to restrict attackers' capabilities and it's important todifferentiate between compromised and suspicious VMs. The countermeasure serves the purpose of 1)protecting the target VMs from being compromised; and2)making attack behavior stand prominent so that the attackers'actions can be identified.

Security Measurement Metrics

The issue of security metrics has attracted much attention and there has been significant effort in the development of quantitative security metrics in recent years. Among different approaches, using attack graph as the security metric model for the evaluation of security risks [28] is a good choice. In order to assess the network security risk condition for the current network configuration, security metrics are needed in the attack graph to measure risk likelihood. After an attack graph is constructed, vulnerability information is included in the graph. For the initial node or external node (i.e., the root of the graph, $NR \subseteq ND$), the *priori probability* is assigned on the likelihood of a threat source becoming active and the difficulty of the vulnerability to be exploited. We use GV to denote the priori risk probability for the root node of the graph and usually the value of GV is assigned to a high probability, e.g., from 0.7 to 1.

For the internal exploitation node, each attack-step node $e \in NC$ will have a probability of vulnerability exploitation denoted as $GM[e]$. $GM[e]$ is assigned according to the Base Score (BS) from CVSS (Common Vulnerability Scoring System). The base score as shown in (1) [24], is calculated by the impact and exploitability factor of the vulnerability. Base score can be directly obtained from National Vulnerability Database [26] by searching for the vulnerability CVE id.

$BS = (0.6 \times IV + 0.4 \times E - 1.5) \times f(IV)$, (1) where,

$IV = 10.41 \times (1 - (1 - C) \times (1 - I) \times (1 - A))$,

$E = 20 \times AC \times AU \times AV$,and

$f(IV) =$ if $IV = 0$,

1.176 otherwise.

The impact value (IV) is computed from three basicparameters of security namely confidentiality (C), integrity(I), and availability (A). The exploitability (E) score consists of access vector (AV), access complexity (AC), and authentication instances (AU). The value of BS ranges from 0 to 10. In our attack graph, we assign each internal node with its BS value divided by 10, as shown in (2).

$GM[e] = Pr(e = T) = BS(e)/10, \forall e \in NC$. (2)

In the attack graph, the relations between exploits can be disjunctive or conjunctive according to howthey are related through their dependency conditions [29]. Such relationships can be represented as *conditionalprobability*, where the risk probability of current node is determined by the relationship with its predecessors and their risk probabilities. We propose the following probability derivation relations:

• for any attack-step node $n \in NC$ with immediate predecessors set $W = parent(n)$,

$Pr(n|W) = GM[n] \times s \in W$

$Pr(s|W)$; (3)

• for any privilege node $n \in ND$ with immediate predecessors set $W = parent(n)$, and then

$Pr(n|W) = 1 - s \in W$

$(1 - Pr(s|W))$. (4)

Once conditional probabilities have been assigned to all internal nodes in SAG, we can merge risk values from all predecessors to obtain the *cumulative risk probability*or *absolute risk probability* for each node according to (5) and (6). Based on derived conditional probability assignments on each node, we can then derive an effective security hardening plan or a mitigation strategy:

• for any attack-step node $n \in NC$ with immediate predecessor set $W = parent(n)$,

$Pr(n) = Pr(n|W) * s \in W$

$Pr(s)$; (5)

• for any privilege node $n \in ND$ with immediate predecessor set $W = parent(n)$,

$Pr(n) = 1 - s \in W$

$(1 - Pr(s))$.(6)

Mitigation Strategies

Based on the security metrics defined in the previoussubsection, NICE is able to construct the mitigation strategies in response to detected alerts. First, we define the term *countermeasure pool* as follows:

Definition (Countermeasure Pool).*A countermeasure*pool CM = {cm1, cm2, . . . ,cmn} is a set of countermeasures. Each cm ∈CM is a tuple cm = (cost, intrusiveness, condition, effectiveness*), where*

1. cost is the unit that describes the expenses required to apply the countermeasure in terms of resources and operational complexity, and it is defined in a range from 1 to 5, and higher metric means higher cost;
2. intrusiveness is the negative effect that a countermeasure brings to the Service Level Agreement (SLA) and its value ranges from the least intrusive (1) to the most intrusive (5), and the value of intrusiveness is 0 if the countermeasure has no impacts on the SLA;
3. condition is the requirement for the corresponding countermeasure;
4. effectiveness is the percentage of probability changes of the node, for which this countermeasure is applied. In general, there are many countermeasures that can be applied to the cloud virtual networking system depending on available countermeasure techniques that can be applied. Without losing the generality, several common virtual-networking-based countermeasures are listed in table 1. The optimal countermeasure selection is a multi-objective optimization problem, to calculate MIN(impact, cost) and MAX(benefit).

In NICE, the network reconfiguration strategies mainly involve two levels of action: layer-2 and layer-

3. At layer-2, virtual bridges (including tunnels that can be established between two bridges) and VLANs

Main component in cloud's virtual networking system toconnect two VMs directly. A virtual bridge is an entity that attaches Virtual Interfaces (VIFs). Virtual machines on different bridges are isolated at layer 2. VIFs on the same virtual bridge but with different VLAN tags cannot communicate to each other directly. Based on this layer-2 isolation, NICE can deploy layer-2 network reconfiguration to isolate suspicious VMs. For example, vulnerabilities due to Arpspoofing [30] attacks are not possible when the suspicious VM is isolated to a different bridge. As a result, this countermeasure disconnects an attack path in the attack graph causing the attacker to explore an alternate attack path. Layer-3 reconfiguration is another way to disconnect an attack path. Through the network controller, the flow table on each OVS or OFS can be modified to change the network topology.

We must note that using the virtual network reconfigurationapproach at lower layer has the advantage in that upper layer applications will experience minimal impact. Especially, this approach is only possible when using software-switching approach to automate the reconfiguration in a highly dynamic networking environment. Countermeasures such as traffic isolation can be implemented by utilizing the traffic engineering capabilities of OVS and OFS to restrict the capacity and reconfigure the virtual network for a suspicious flow. When a suspicious activity such as network and port scanning is detected in the cloud system, it is important

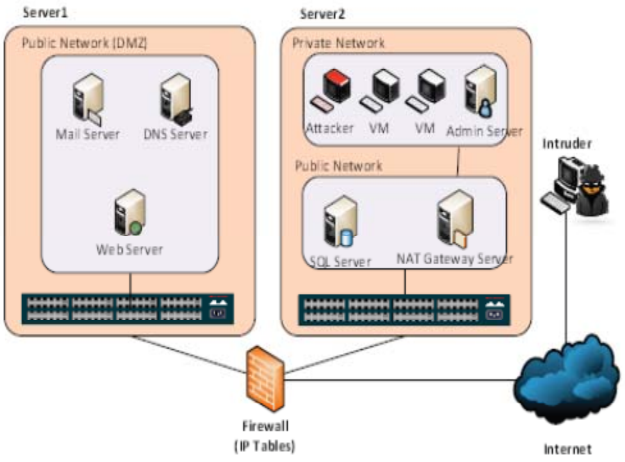
to determine whether the detected activity is indeed malicious or not. For example, attackers can purposely hide their scanning behavior to prevent the NIDS from identifying their actions. In such situation, changing the network configuration will force the attacker to perform more explorations, and in turn will make their attacking behavior stand out.

Countermeasure selection

Algorithm 2 presents how to select the optimal countermeasure for a given attack scenario. Input to the algorithm is an *alert*, attack graph *G*, and a pool of countermeasures *CM*. The algorithm starts by selecting the node *vAlert* that corresponds to the alert generated by a NICE-A. Before selecting the countermeasure, wecount the distance of *vAlert* to the *target node*. If the distance is greater than a threshold value, we do not perform countermeasure selection but update the ACG to keep track of alerts in the system (line 3). For the source node *vAlert*, all the reachable nodes (including the source node) are collected into a set *T* (line 6). Because the alert is generated only after the attacker has performed the action, we set the probability of *vAlert*to 1 and calculate the new probabilities for all of its child (downstream) nodes in the set *T* (line 7 & 8). Now for all $t \in T$ the applicable countermeasures in *CM* are selected and new probabilities are calculated according to theeffectiveness of the selected countermeasures (line 13 &14). The change in probability of *target node* gives the *benefit* for the applied countermeasure using (7). In the next double for-loop, we compute the Return of Investment (ROI) for each *benefit* of the applied countermeasure based on (8). The countermeasure which when applied on a node gives the least value of ROI, is regarded as the optimal countermeasure. Finally, SAG and ACG are also updated before terminating the algorithm. The complexity of Algorithm 2 is $O(|V| \times |CM|)$ where $|V|$ is the number of vulnerabilities and $|CM|$ represents the number of countermeasures.

Algorithm 2 Countermeasure Selection

```
Require: Alert, G(E, V), CM
1: Let vAlert = Source node of the Alert
2: if Distance to Target(vAlert) > threshold then
3:     Update ACG
4:     return
5: end if
6: Let T = Descendant(vAlert) ∪ vAlert
7: Set Pr(vAlert) = 1
8: Calculate Risk Prob(T)
9: Let benefit[|T|, |CM|] = Ø
10: for each t ∈ T do
11:     for each cm ∈ CM do
12:         if cm.condition(t) then
13:             Pr(t) = Pr(t) ∗ (1 − cm.effectiveness)
14:             Calculate Risk Prob(Descendant(t))
15:             benefit[t, cm] = ΔPr(target node). (7)
16:         end if
17:     end for
18: end for
19: Let ROI[|T|, |CM|] = Ø
20: for each t ∈ T do
21:     for each cm ∈ CM do
22:
        ROI[t, cm] =
        benefit[t, cm]
        cost.cm + intrusiveness.cm. (8)
23:     end for
24: end for
25: Update SAG and Update ACG
26: return Select Optimal CM(ROI)
```



VII. PERFORMANCE EVALUATION

In this section we present the performance evaluation ofNICE. Our evaluation is conducted in two directions: the security performance, and the system computing and network reconfiguration overhead due to introduced security mechanism.

Security Performance Analysis

To demonstrate the security performance of NICE, wecreated a virtual network testing environment consisting of all the presented components of NICE.

Environment and Configuration

To evaluate the security performance, a demonstrative virtual cloud system consisting of public (public virtual servers) and private (VMs) virtual domains is established as shown in Figure 3. Cloud Servers 1 and 2 are connected to Internet through the external firewall. In the Demilitarized Zone (DMZ) on Server 1, there is one Mail server, one DNS server and one Web server. Public network on Server 2 houses SQL server and NAT Gateway Server. Remote access to VMs in the private network is controlled through SSHD (i.e., SSH Daemon) from the NAT Gateway Server. Table 2 shows the vulnerabilities present in this network and table 3 shows the corresponding network connectivity that can be explored based on the identified vulnerabilities. *Attack Graph and Alert Correlation*The attack graph can be generated by utilizing network topology and the vulnerability information, and it is shown in Figure 4. As the attack progresses, the system generates various alerts that can be related to the nodes in the attack graph.

Creating an attack graph requires knowledge of network connectivity, running services and their vulnerability information. This information is provided to the attack graph generator as the input. Whenever a new vulnerability is discovered or there are changes in the network connectivity and services running through them, the updated information is provided to attack graph generator and old attack graph is updated to a new one. *SAG* provides information about the possible paths that an attacker can follow. *ACG* serves the purpose of confirming attackers' behavior, and helps in determining false positive and false negative. *ACG* can also be helpful in predicting attackers' next steps.

Countermeasure Selection

To illustrate how NICE works, let us consider for example,an alert is generated for node 16 (*vAlert*= 16) when the system detects LICQ Buffer overflow. After the alert is generated, the cumulative probability of node 16 becomes 1 because that attacker has already compromised that node. This triggers a change in cumulative probabilities of child nodes of node 16. Now the next step is to select the countermeasures from the pool of countermeasures *CM*. If the countermeasure CM4: create filtering rules is applied to node 5 and we assume that this countermeasure has effectiveness of85%, the probability of node 5 will change to 0.1164, which causes change in probability values of all child nodes of node 5 thereby accumulating to a decrease of 28.5% for the target node 1. Following the same approach for all possible countermeasures that can be applied, the percentage change in the cumulative probability of node 1, i.e., *benefit* computed using (7) are shown in Figure 5. Apart from calculating the *benefit* measurements, we also present the evaluation based on *Return of Investment(ROI)* using (8) and represent a comprehensive evaluation considering *benefit*, *cost* and *intrusiveness* of countermeasure. Figure 6 shows the *ROI* evaluations for presented countermeasures. Results show that countermeasures CM2 andCM8 on node 5 have the maximum *benefit* evaluation, however their cost and intrusiveness scores indicate that they might not be good candidates for the optimal countermeasure and *ROI* evaluation results confirm this. The *ROI* evaluations demonstrate that CM4 on node 5 is the optimal solution.

VIII.Conclusion And Future Work

In this paper, we presented NICE, which is proposed to detect and mitigate collaborative attacks in the cloud virtual networking environment. NICE utilizes the attack graph model to conduct attack detection and prediction. The proposed solution investigates how to use the programmability of software switches based solutions to improve the detection accuracy and defeat victim exploitation phases of collaborative attacks. The system performance evaluation demonstrates the feasibility of NICE and shows that the proposed solution can significantly reduce the risk of the cloud system from being exploited and abused by internal and external attackers. NICE only investigates the network IDS approach to counter zombie explorative attacks. In order to improve the detection accuracy, host-based IDS solutions are needed to be incorporated and to cover the whole spectrum of IDS in the cloud system. This should be investigated in the future work. Additionally, as indicated in the paper, we will investigate the scalability of the proposed NICE solution by investigating the decentralized network control and attack analysis model based on current study.

References

[1] Coud Sercurity Alliance, "Top threats to cloud computing v1.0,"https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf, March 2010.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia "A view of cloud computing," *ACM Commun.*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[3] B. Joshi, A. Vijayan, and B. Joshi, "Securing cloud computing environment against DDoS attacks," *IEEE Int'l Conf. Computer Communication and Informatics (ICCCI '12)*, Jan. 2012.

[4] H. Takabi, J. B. Joshi, and G. Ahn, "Security and privacy challenges in cloud computing environments," *IEEE Security & Privacy*, vol. 8, no. 6, pp. 24–31, Dec. 2010.

[5] "Open vSwitch project," http://openvswitch.org, May 2012.

[6] Z. Duan, P. Chen, F. Sanchez, Y. Dong, M. Stephenson, and J. Barker, "Detecting spam zombies by monitoring outgoing messages," *IEEE Trans. Dependable and Secure Computing*, vol. 9, no. 2, pp. 198–210, Apr. 2012. IEEE TRANSACTIONS ON DEPEDABLE AND SECURE COMPUTING

[7] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "BotHunter: detecting malware infection through IDS-driven dialog correlation," *Proc. of 16th USENIX Security Symp. (SS '07)*, pp. 12:1–12:16, Aug. 2007.

[8] G. Gu, J. Zhang, and W. Lee, "BotSniffer: detecting botnet command and control channels in network traffic," *Proc. of 15th Ann. Network and Distributed Sytem Security Symp. (NDSS '08)*, Feb. 2008.

[9] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," *Proc. IEEE Symp. on Security and Privacy*, 2002, pp. 273–284.

[10]"NuSMV: A new symbolic model checker," http://afrodite.itc.it:1024/~numb. Aug. 2012.

[11] S. H. Ahmadinejad, S. Jalili, and M. Abadi, "A hybrid model for correlating alerts of known and unknown attack scenarios and updating attack graphs," *Computer Networks*, vol. 55, no. 9, pp. 2221–2240, Jun. 2011.

[12] S. Roschke, F. Cheng, and C. Meinel, "A new alert correlation algorithm based on attack graph," *Computational Intelligence in Security for Information Systems*, LNCS, vol. 6694, pp. 58–67. Springer, 2011.

[15] A. Roy, D. S. Kim, and K. Trivedi, "Scalable optimal countermeasure selection using implicit enumeration on attack countermeasure trees," *Proc. IEEE Int'l Conf. on Dependable Systems Networks (DSN '12)*, Jun. 2012.

[16] "Open flow." http://www.openflow.org/wp/learnmore/, 2012.