ICE: Intrusion Detection and Countermeasure Selection in Virtual Network Systems

1. **Ashwini B Korwar,** PG Student ,Aryabhata Institute of Technology & Science, Hyderabad

2. **Santosh Kumar** ,Assistant Professor, Aryabhata Institute of Technology & Science,Hyderabad

Abstract—Cloud security is one of most important issues that has attracted a lot of research and development effort in past few years. Particularly, attackers can explore vulnerabilities of a cloud system and compromise virtual machines to deploy further large-scale Distributed Denial-of-Service (DDoS). DDoS attacks usually involve early stage actions such as multistep exploitation, low-frequency vulnerability scanning, and compromising identified vulnerable virtual machines as zombies, and finally DDoS attacks through the compromised zombies. Within the cloud system, especially the Infrastructure-as-a-Service (IaaS) clouds, the detection of zombie exploration attacks is extremely difficult. This is because cloud users may install vulnerable applications on their virtual machines. To prevent vulnerable virtual machines from being compromised in the cloud, we propose a multiphase distributed vulnerability detection, measurement, and countermeasure selection mechanism called ICE, which is built on attack graph-based analytical models and reconfigurable virtual network-based countermeasures. The proposed framework leverages OpenFlow network programming APIs to build a monitor and control plane over distributed programmable virtual switches to significantly improve attack detection and mitigate attack consequences. The system and security evaluations demonstrate the efficiency and effectiveness of the proposed solution.

Key words—Network security, cloud computing, intrusion detection, attack graph, zombie detection

1 INTRODUCTION

Recent studies have shown that users migrating to the cloud Security Alliance (CSA) survey shows that among all security issues, abuse and nefarious use of cloud computing is considered as the top security threat [1], in which attackers can exploit vulnerabilities in clouds and utilize cloud system resources to deploy attacks. In traditional data centers, where system administrators have full control over the host machines, vulnerabilities can be detected and patched by the system administrator in a centralized manner. However, patching known security holes in cloud data centers, where cloud users usually have the privilege to control software installed on their managed VMs, may not work effectively and can violate the service level agreement (SLA). Furthermore, 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. 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 [3]. Such attacks are more effective in the cloud environment because 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[4].Intrusion detection and Countermeasure selection in virtual network systems (ICE) to establish a defense-in-depth intrusion detection framework. For better attack detection, ICE incorporates attack graph analytical procedures into the intrusion detection processes. We must note that the design of ICE does not intend to improve any of the existing intrusion detection algorithms; indeed, ICE employs a reconfigurable virtual networking approach to detect and counter the attempts to compromise VMs, thus preventing zombie VMs. In general, ICE includes two main phases: 1) deploy a lightweight mirroring-based network intrusion detection agent (ICE-A) on each cloud server to capture and analyze cloud traffic. A ICE-A periodically scans the virtual system vulnerabilities within a cloud server to establish Scenario Attack Graph (SAGs), and then based on the severity of identified vulnerability toward the collaborative attack goals, ICE will decide whether or not to put a VM in network inspection state. 2) Once a VM enters inspection state, Deep Packet Inspection (DPI) is applied, and/or virtual network reconfigurations can be deployed to the inspecting VM to make the potential attack behaviors prominent.

ICE significantly advances the current network IDS/ IPS solutions by employing programmable virtual networking approach that allows the system to construct a dynamic reconfigurable IDS system. By using software switching techniques [5], ICE constructs a mirroring-based traffic capturing framework to minimize the interference on users' traffic compared to traditional bump-in-the-wire (i.e., proxy-based) IDS/IPS. The programmable virtual networking architecture of ICE enables the cloud to establish inspection and quarantine modes for suspicious VMs according to their current vulnerability state in the current SAG. Based on the collective behavior of VMs in the SAG, ICE can decide appropriate actions, for example, DPI or traffic filtering, on the suspicious VMs. Using this approach, ICE does not need to block traffic flows of a suspicious VM in its early attack stage. The contributions of ICE are presented as follows:

- We devise ICE, a new multiphase distributed network intrusion detection and prevention frame.
- work in a virtual networking environment that captures and inspects suspicious cloud traffic without interrupting users' applications and cloud services.
- ICE incorporates a software switching solution to quarantine and inspect suspicious VMs for further investigation and protection. Through programmable network approaches, ICE can improve the attack detection probability and improve the resiliency to VM exploitation attack without interrupting existing normal cloud services.
- ICE employs a novel attack graph approach for attack detection and prevention by correlating attack behavior and also suggests effective counter- measures.
- ICE optimizes the implementation on cloud servers to minimize resource consumption. Our study shows that ICE consumes less computational overhead compared to proxy-based network intrusion detection solutions.

2 ICE MODELS

In this section, we describe how to utilize attack graphs to model security threats and vulnerabilities in a virtual networked system, and propose a VM protection model based on virtual network reconfiguration approaches to prevent VMs from being exploited.

2.1 Threat Model

In our attack model, we assume that an attacker can be located either outside or inside of the virtual networking system. The attacker's primary goal is to exploit vulnerable VMs and compromise them as zombies. Our protection model focuses on virtual-network-based attack detection and reconfiguration solutions to improve the resiliency to zombie explorations. Our work does not involve host-based IDS and does not address how to handle encrypted traffic for attack detections. Our proposed solution can be deployed in an Infra-structure-as-a-Service (IaaS) cloud networking system, and we assume that the Cloud Service Provider (CSP) is benign. We also assume that cloud service users are free to install whatever operating systems or applications they want, even if such action may introduce vulnerabilities to their controlled VMs. Physical security of cloud server is out of scope of this paper. We assume that the hypervisor is secure and free of any vulnerabilities.

2.2 Attack Grahp Model

An attack graph is a modeling tool to illustrate all possible multistage, multihost attack paths that are crucial to understand threats and then to decide appropriate counter- measures [14]. In an attack graph, each node represents either precondition or consequence of an exploit. The actions are not necessarily an active attack because normal protocol interactions can also be used for attacks. Attack graph is helpful in identifying potential threats, possible attacks, and known vulnerabilities in a cloud system. Since the attack graph provides details of all known vulnerabilities in the system and the connectivity information, we get a whole picture of current security situation of the system, where we can predict the possible threats and attacks by correlating detected events or activities. If an event is recognized as a potential attack, we can apply specific countermeasures to mitigate its impact or take actions to prevent it from contaminating the cloud system. To represent the attack and the result of such actions, we extend the notation of MulVAL logic attack graph as presented by Ou et al. [8] and define as Scenario Attack Graph (SAG).

Definition 1 (SAG). An SAG is a tuple SAG $=(V, E)$, where

1. $V = N_C \cup N_D \cup N_R$ denotes a set of vertices that include three types namely conjunction node N_C to represent exploit, disjunction node N_D to denote result of exploit, and root node N_R for showing initial step of an attack scenario.
2. $E = E_{pre} \cup E_{post}$ denotes the set of directed edges. An edge $e \in E_{pre} \subseteq N_D \times N_C$ represents that N_D must be satisfied to achieve N_C. An edge $e \in E_{post} \subseteq N_C \times N_D$ means that the consequence shown by N_D can be obtained if N_C is satisfied.

Node $v_C \in N_C$ is defined as a three tuple (Hosts, vul, alert) representing a set of IP addresses, vulnerability information such as CVE [15]and alerts related to v_C, respectively. N_D behaves like a logical OR operation and contains details of the results of actions. N_R represents the root node of the SAG.

For correlating the alerts, we refer to the approach described in [15] and define a new Alert Correlation Graph (ACG) to map alerts in ACG to their respective nodes in SAG. To keep track of attack progress, we track the source and destination IP addresses for attack activities.

Definition 2 (ACG). An ACG is a three tuple ACG $=(A, E, P)$, where

1. A is a set of aggregated alerts. An alert a 2 A is a data structure (src, dst, cls, ts) representing source IP address, destination IP address, type of the alert, and time stamp of the alert respectively.

2. Each alert a maps to a pair of vertices vc, vd in SAG using map(a) function, i.e.,
$map(a): a \mapsto \{(vc, vd) | (a.src \in vc.Hosts) \wedge (a.dst \in vd.Hosts) \wedge (a.cls = vc.vul)\}$

3. E is a set of directed edges representing correlation between two alerts (a, a') if

criteria below satisfied:

a. $(a.ts < a'.ts) \wedge (a'.ts - a.ts < threshold)$.

b. $\exists(vd, vc) \in Epre: (a.dst \in vd.Hosts \wedge \quad a'.src \in vc.Hosts)$.

4. P is set of paths in ACG. A path $Si \subset P$ is a set of related alerts in chronological order.

We assume that A contains aggregated alerts rather than raw alerts. Raw alerts having same source and destination IP addresses, attack type, and time stamp within a specified window are aggregated as Meta Alerts. Each ordered pair (a, a') in ACG maps to two neighbor vertices in SAG with time stamp difference of two alerts within a predefined threshold. ACG shows dependency of alerts in chronological order and we can find related alerts in the same attack scenario by searching the alert path in ACG. A set P is used to store all paths from root alert to the target alert in the SAG, and each path S_i P represents alerts that belong to the same attack scenario.

Algorithm 1. Alert_Correlation

Require: alert a_c, SAG, ACG

```
if (ac is a new alert) then
    create node ac in ACG
    n1 ←vc ϵ map ( ac )
    for all n2 ϵ parent(n1) do
        create edge (n2 .alert,ac )
        for all Si containing a do
            if a is the last element in Si then
                append ac to Si
            else
                create path Si + 1 = {subset(Si, a), ac}
            end if
        end for
        add ac to n1.alert
    end for
end if
return S
```

2.3 VM Protection Model

The VM protection model of ICE consists of a VM profiler, a security indexer, and a state monitor. We specify security index for all the VMs in the network depending upon various factors like connectivity, the number of vulnerabilities present and their impact scores. The impact score of a vulnerability, as defined by the CVSS guide [16], helps to judge the confidentiality, integrity, and availability impact of the vulnerability being exploited. Connectivity metric of a VM is decided by evaluating incoming and outgoing connections.

Definition 3 (VM State). Based on the information gathered from the network controller, VM states can be defined as following:

1. Stable.
2. Vulnerable.
3. Exploited.
4. Zombie.

3 ICE SYSTEM DESIGN

In this section, we first present the system design overview of ICE and then detailed descriptions of its components.

3.1 System Design Overview

The proposed ICE framework is illustrated in Fig. 1. It shows the ICE framework within one cloud server cluster. Major components in this framework are distributed and light-weighted ICE-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). ICE-A 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 OpenFlow 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. ICE-A is a network intrusion detection engine that can be installed in either Dom0 or DomU 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 OFSs) or when new alerts are generated by the ICE-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 countermeasure actions by reconfiguring virtual or physical OFSs.

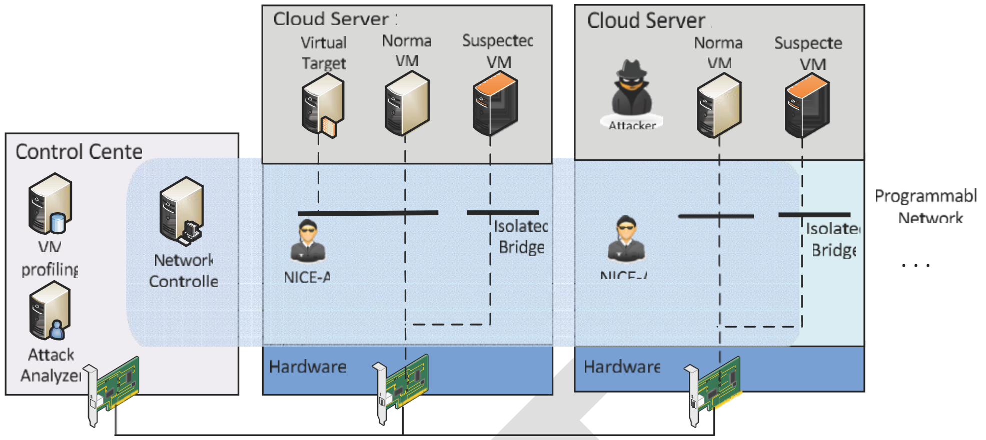

Fig. 1. ICE architecture within one cloud server cluster.

3.2.1 ICE-A

The ICE-A is a Network-based Intrusion Detection System (NIDS) agent installed in either Dom0 or DomU in each cloud server. It scans the traffic going through Linux bridges that control all the traffic among VMs and in/out from the physical cloud servers. In our experiment, Snort is used to implement ICE-A in Dom0. It will sniff a mirroring port on each virtual bridge in the Open vSwitch (OVS). 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. The ICE-A sniffing rules have been custom defined to suite our needs. Dom0 in the Xen environment is a privilege domain, that includes a virtual switch for traffic switching among VMs and network drivers for physical network interface of the cloud server. It is more efficient to scan the traffic in Dom0 because all traffic in the cloud server needs go through it; however, our design is independent to the installed VM.

The individual alert detection's false alarm rate does not change. However, the false alarm rate could be reduced through our architecture design. We will discuss more about this issue in the later section.

3.2.2 VM Profiling

Virtual machines in the cloud can be profiled to get precise information about their state, services running, open ports, and so on. One major factor that counts toward a VM profile is its connectivity with other VMs. Any VM that is connected to more number of machines is more crucia than the one connected to fewer VMs because the effect of compromise of a highly connected VM can cause more damage. 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 actors combined will form the VM profile.VM profiles are maintained in a database and contain comprehensive information about vulnerabilities, alert, and traffic. The data comes from:

- Attack graph generator. While generating the attack graph, every detected vulnerability is added to its corresponding VM entry in the database.
- ICE-A. The alert involving the VM will be recorded in the VM profile database.
- Network controller. The traffic patterns involving the VM are based on five tuples (source MAC address, destination MAC address, source IP address, desti- nation IP address, protocol).

3.2.3 Attack Analyzer

The major functions of ICE 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 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. Each node in the attack graph represents an exploit by the attacker. Each path from an initial node to a goal node represents a successful attack.

In summary, ICE attack graph is constructed based on the following information:

- Cloud system information is collected from the node controller (i.e., Dom0 in XenServer). The information includes the number of VMs in the cloud server, running services on each VM, and VM's Virtual Interface (VIF) information.
- Virtual network topology and configuration information is collected from the network controller, which includes virtual network topology, host connectivity, VM connectivity, every VM's IP address, MAC address, port information, and traffic flow information.
- Vulnerability information is generated by both on demand vulnerability scanning (i.e., initiated by the network controller and ICE-A) and regular penetration testing using the well-known vulnerability databases, such as Open Source Vulnerability Data- base (OSVDB) [17], Common Vulnerabilities and Exposures List (CVE) [15], and NIST National Vulnerability Database (NVD) [18].

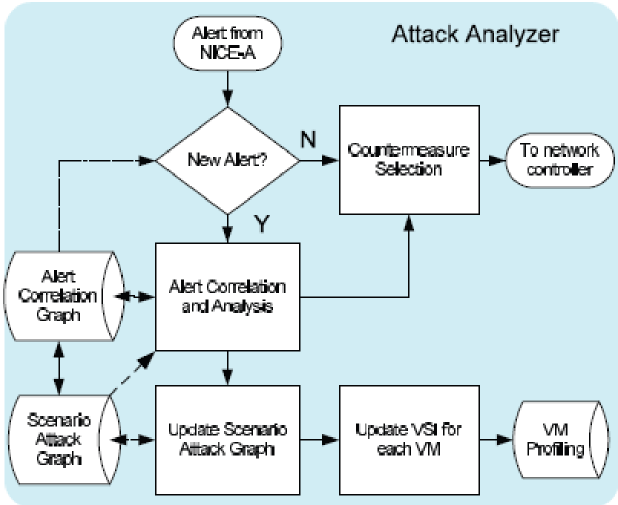

Fig. 2. Workflow of attack analyzer.

The attack analyzer also handles alert correlation and analysis operations. This component has two major functions: 1) constructs ACG, and 2) provides threat information and appropriate countermeasures to network controller for virtual network reconfiguration. Fig. 2 shows the workflow in the attack analyzer component. After receiving an alert from ICE-A, alert analyzer matches the alert in the ACG. If the alert already exists in the graph and it is a known attack (i.e., matching the attack signature), the attack analyzer performs counter- measure selection procedure according to the algorithm described in Section 5.3. and then notifies network controller immediately to deploy countermeasure or mitigation actions. If the alert is new, attack analyzer will perform alert correlation and analysis according to Algorithm 1, and updates ACG and SAG.

This algorithm correlates each new alert to a matching alert correlation set (i.e., in the same attack scenario). A selected countermeasure is applied by the network controller based on the severity of evaluation results. If the alert is a new vulnerability and is not present in the ICE attack graph, the attack analyzer adds it to attack graph and then reconstructs it.

3.2.4 Network Controller

The network controller is a key component to support the programmable networking capability to realize the virtual network reconfiguration feature based on OpenFlow protocol [20]. In ICE, within each cloud server there is a software switch, for example, OVS [5], which is used as the edge switch for VMs to handle traffic in and out from VMs. The communication between cloud servers (i.e., physical servers) is handled by physical OpenFlow-capable Switch (OFS). In ICE, 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 OpenFlow network and provides input to the attack analyzer to construct attack graphs. Through the cloud internal discovery modules that use DNS, DHCP, LLDP, and flow initiations [19], network controller is able to discover the network connectivity information from OVS and OFS. This information includes current data paths on each switch and detailed flow information associated with these paths, such as TCP/IP and MAC header.

The network flow and topology change information will be automatically sent to the controller and then delivered to attack analyzer to reconstruct attack graphs. Another important function of the network controller is to assist the attack analyzer module. According to the OpenFlow protocol [12], when the controller receives the first packet of a flow, it holds the packet and checks the flow table for complying traffic policies. In ICE, 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. Once a traffic flow is admitted, the following packets of the flow are not handled by the network controller, but monitored by the ICE-A.

Network controller is also responsible for applying the countermeasure from attack analyzer. Based on VM Security Index (VSI) and severity of an alert, countermeasures are selected by ICE and executed by the network controller. If a severe alert is triggered and identifies some known attacks, or a VM is detected as a zombie, the network controller will block the VM immediately. An alert with medium threat level is triggered by a suspicious compromised VM.

Countermeasure in such case is to put the suspicious VM with exploited state into quarantine mode and redirect all its flows to ICE-A DPI mode. An alert with a minor threat level can be generated due to the presence of a vulnerable VM. For this case, to intercept the VM's normal traffic, suspicious traffic to/from the VM will be put into inspection mode, in which actions such as restricting its flow bandwidth and changing network configurations will be taken to force attack exploration behavior to stand out.

4 ICE SECURITY MEASUREMENT ATTACK MITIGATION, AND COUNTERMEASURES

In this section, we present the methods for selecting the countermeasures for a given attack scenario. When vulnerabilities are discovered or some VMs are identified as suspicious, several countermeasures can be taken to restrict attackers' capabilities and it is important to differentiate between compromised and suspicious VMs. The countermeasure serves the purpose of: 1) protecting the target VMs from being compromised, and 2) making attack behavior stand prominent so that the attackers' actions can be identified.

4.1 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 [20] is a good choice. 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, $N_R \subseteq N_D$).), 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 G_V to denote the priori risk probability for the root node of the graph and usually the value of G_V is assigned to a high probability, e.g., from 0.7 to 1.

For the internal exploitation node, each attack-step node $e \in N_C$ will have a probability of vulnerability exploitation denoted as G_M [e] . G_M [e] is assigned according to the Base Score (BS) from Common Vulnerability Scoring System (CVSS). The BS as shown in (1) [16] is calculated by the impact and exploitability factor of the vulnerability. BS can be directly obtained from National Vulnerability Database [18] by searching for the vulnerability CVE id

$$BS=(0:6\times IV+0.4\times E-1.5\times f(IV), \qquad (1)$$

Where

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

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

and

$f(IV) = \{ 0$ if $IV = 0$, 1.176 otherwise. The impact value (IV) is computed from three basic parameters 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

$$GM[e] = \Pr(e = T) = \frac{BS(e)}{10}, \forall e \in Nc$$

In the attack graph, the relations between exploits can be disjunctive or conjunctive according to how they are related through their dependency conditions [21]. Such relationships can be represented as conditional probability, 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 $n \in N_C$ with immediate predecessors set $W = parent(n)$,

$$Pr(n|W) = G_M[n] \times \prod_{s \in W} Pr(s|W); \qquad (3)$$

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

$$Pr(n|W) = 1 - \prod_{s \in W} (1 - Pr(s|W)). \qquad (4)$$

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 $n \in N_C$ with immediate predecessor set $W = parent(n)$,

$$Pr(n) = Pr(n|W) \times \prod_{s \in W} Pr(s); \qquad (5)$$

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

$$Pr(n) = 1 - \prod_{s \in W} (1 - Pr(s)). \qquad (6)$$

4.2 Mitigation Strategies

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

Definition 4 (Countermeasure Pool). A countermeasure pool $CM = \{cm_1, cm_2, \ldots, cm_n\}$ is a set of countermeasures.

Each $cm \in CM$ 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 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).

TABLE 1
Possible Countermeasure Types

No.	Countermeasure	Intrusiveness	Cost
1	Traffic redirection	3	3
2	Traffic isolation	4	2
3	Deep Packet Inspection	3	3
4	Creating filtering rules	1	2
5	MAC address change	2	1
6	IP address change	2	1
7	Block port	4	1
8	Software patch	5	4
9	Quarantine	5	2
10	Network reconfiguration	0	5
11	Network topology change	0	5

In ICE, 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 are main component in cloud's virtual networking system to connect two VMs directly. A virtual bridge is an entity that attaches 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, ICE can deploy layer-2 network reconfiguration to isolate suspicious VMs. For example, vulnerabilities due to Arpspoofing [22] 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 reconfiguration approach 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 Update_SAG and Update_ACG out.

4.3 Countermeasure Selection

counter- measure for a given attack scenario. Input to the algorithm is an alert, attack graph G, and a pool of countermeasures CM.

Algorithm 2. Countermeasure_Selection

Require: Alert, G(E, V), CM

Let v_{Alert} = Source node of the Alert
if Distance to Target(v_{Alert}) > threshold then
  Update ACG
  return
end if
Let T = Descendant(v_{Alert}) ∪ v_{Alert}
Set Pr(v_{Alert}) = 1
Calculate_Risk_Prob(T)
Let benefit [|T|,|CM|] = ∅
for each t ∈ T do
  for each cm ∈ CM do
    if cm.condition(t) then
      Pr(t) = Pr(t) *(1-cm.effectiveness)
      Calculate_Risk_Prob(Descendant(t))
  benefit[t, cm] = ΔPr(target node). (7)
    end if
  end for
end for
Let ROI[|T|,|CM|] = ∅
for each t ∈ T do
 for each cm ∈ CM do
$ROI[t, cm] = \frac{benfit\ [t,cm]}{cost}.cm + intrusiveness.cm$
(8)
  end for
end for
Update_SAG and Update_ACG
returne Select_Optimal _CM(ROI)

5 CONCLUSION

The technique of ICE, is proposed to detect and mitigate collaborative attacks in the cloud virtual networking environment. ICE utilizes the attack graph model to conduct attack detection and prediction. The proposed solution investigates how to use the program- mobility 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 ICE 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. ICE only investigates the network IDS approach to counter zombie explorative attacks. 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.

REFERENCES

[1] Coud Sercurity Alliance, “Top Threats to Cloud Computing v1.0,” https://cloudsecurityalliance.org/topthreats/csathreats. v1.0.pdf, Mar. 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 Comm., vol. 53, no. 4, pp. 50-58, Apr. 2010.
[3] B. Joshi, A. Vijayan, and B. Joshi, “Securing Cloud Computing Environment Against DDoS Attacks,” Proc. IEEE Int’l Conf. Computer Comm. 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 and Privacy, vol. 8, no. 6, pp. 24-31, Dec. 2010.
[5] “Open vSwitch Project,” http://openvswitch.org, May 2012.
[6] G. Gu, J. Zhang, and W. Lee, “BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic,” Proc. 15th Ann. Network and Distributed Sytem Security Symp. (NDSS ’08), Feb.2008.
[7] P. Ammann, D. Wijesekera, and S. Kaushik, “Scalable, graph- based network vulnerability analysis,” Proc. 9th ACM Conf. Computer and Comm. Security (CCS ’02), pp. 217-224, 2002.
[8] X. Ou, S. Govindavajhala, and A.W. Appel, “MulVAL: A Logic- Based Network Security Analyzer,” Proc. 14th USENIX Security Symp., pp. 113-128, 2005.
[9] R. Sadoddin and A. Ghorbani, “Alert Correlation Survey: Frame- work and Techniques,” Proc. ACM Int’l Conf. Privacy, Security and Trust: Bridge the Gap between PST Technologies and Business Services (PST ’06), pp. 37:1-37:10, 2006.
[10] L. Wang, A. Liu, and S. Jajodia, “Using Attack Graphs for Correlating, Hypothesizing, and Predicting Intrusion Alerts,” Computer Comm., vol. 29, no. 15, pp. 2917-2933, Sept. 2006.

[11] S. Roschke, F. Cheng, and C. Meinel, "A New Alert Correlation Algorithm Based on Attack Graph," Proc. Fourth Int'l Conf. Computational Intelligence in Security for Information Systems, pp. 58-67, 2011.
[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," SIGCOMM Computer Comm. Rev., vol. 38, no. 2, pp. 69-74, Mar. 2008.
[13] E. Keller, J. Szefer, J. Rexford, and R.B. Lee, "NoHype: Virtualized Cloud Infrastructure without the Virtualization," Proc. 37th ACM Ann. Int'l Symp. Computer Architecture (ISCA'10), pp. 350-361, June 2010.
[14] X. Ou, W.F. Boyer, and M.A. McQueen, " A Scalable Approach to Attack Graph Generation," Proc. 13th ACM Conf. Computer and Comm. Security (CCS '06), pp. 336-345, 2006.
[15] Mitre Corporation, "Common Vulnerabilities and Exposures, CVE," http://cve.mitre.org/, 2012.
[16] P. Mell, K. Scarfone, and S. Romanosky, "Common Vulnerability Scoring System (CVSS)," http://www.first.org/cvss/cvss-guide. html, May 2010.
[17] O. Database, "Open Source Vulnerability Database (OVSDB)," http://osvdb.org/, 2012.
[18] National Institute of Standards and Technology, "National Vulnerability Database, NVD," http://nvd.nist. gov, 2012.
[19] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an Operating System for Networks," SIGCOMM Computer Comm. Rev., vol. 38, no. 3,pp. 105-110, July 2008.
[20] X. Ou and A. Singhal, Quantitative Security Risk Assessment Of Enterprise Networks. Springer, Nov. 2011.
[21] M. Frigault and L. Wang, "Measuring Network Security Using Bayesian Network-Based Attack Graphs," Proc. IEEE 32nd Ann. Int'l Conf. Computer Software and Applications (COMPSAC '08),pp. 698-703, Aug. 2008.
[22] K. Kwon, S. Ahn, and J. Chung, "Network Security Management Using ARP Spoofing," Proc. Int'l Conf. Computational Science and Its Applications (ICCSA '04), pp. 142-149, 2004.
[23] "Metasploit," http://www.metasploit.com,2012.
[24] "Armitage," http://www.fastandeasyhacking.com,2012.
[25] M. Tupper and A. Zincir-Heywood, "VEA-bility Security Metric: A Network Security Analysis Tool," Proc. IEEE Third Int'l Conf.Availability, Reliability and Security (ARES '08), pp.950-957, Mar.2008.