International Journal of Computer Science
Oddity...Probe...Reviste...
http://www.ijcsjournal.com Volume 2, Issue 2, No 1, 2014. ISSN: 2348-6600.
Reference ID: IJCS-054. PAGE NO: 311-319.

Intrusion Detection and Preventing Attacks using Sequential Algorithm

R.Sangeethapriya

Department of Computer Science
Bharathiyar Institution of Engineering for Women
Salem, Tamilnadu, India
Priyasangeetha20@gmail.com

Abstract *: Cloud security has attack third-party storage services in outsources data backups reduce data management costs. DDOS attack through compromised zombies. IDS secure overlay cloud storage system achieves policy-based, access control, file assured deletion. IDS built upon cryptographic Key operations that are independent of third-party clouds. It associates outsourced files with file access policy, assured deletes file make them unrecoverable anyone revocations using file access policies. While, introducing IDS only minimal performance and monetary cost overhead. Insights provide system and security evaluations demonstrate efficiency and effectiveness into today's cloud storage services.*

Keywords: *Distributed Denial of Service, cloud Security Alliance, Open Flow, Scenario attack graph, zombie detection, Intrusion Prevention System.*

I. INTRODUCTION

A recent 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, 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 loop holes 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. Here 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. 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.

International Journal of Computer Science
Oddity...Probe...Reviste...
http://www.ijcsjournal.com Volume 2, Issue 2, No 1, 2014. ISSN: 2348-6600.
Reference ID: IJCS-054. PAGE NO: 311-319.

NICE 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, NICE constructs a mirroring based traffic capturing framework to minimize the interference on users' traffic compared to traditional bum in the wire IDS/IPS. The programmable virtual networking architecture of NICE enables the cloud to establish inspection and quarantine modes for suspicious VMs according to their current vulnerability state in the current SAG. NICE, which is 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, also investigate the scalability of the proposed NICE solution by investigating the decentralized network control and attack analysis model based on current study.

II. Related Works

In this section, present literatures of several highly related research areas to NICE, including: Zombie detection and prevention, attack graph construction and security analysis, and software defined networks for attack countermeasures.

The area of detecting malicious behavior has been well explored. The work by Duan et al. [6] focuses on the detection of compromised machines that have been recruited to serve as spam zombies. Their approach, SPOT, is based on sequentially scanning outgoing messages while employing a statistical method Sequential Probability Ratio Test (SPRT), to quickly determine whether a host has been compromised. BotHunter [7] detects compromised machines based on the fact that a thorough malware infection process has a number of well-defined stages that allow correlating the intrusion alarms triggered by inbound traffic with resulting outgoing communication patterns. BotSniffer [8] exploits uniform spatial-temporal behavior characteristics of compromised machines to detect zombies by grouping flows according to server connections and searching for similar behavior in the flow.

An attack graph is able to represent a series of exploits, called atomic, attacks that lead to an undesirable state, for example, a state where an attacker has obtained administrative access to a machine. There are many automation tools to construct attack graph. Sheyner et al. [9] proposed a technique based on a modified symbolic model checking NuSMV [10] and Binary Decision Diagrams (BDDs) to construct attack graph. Their model can generate all possible attack paths; however, the scalability is a big issue for this solution. P. Amman et al. [11] introduced the assumption of monotonicity, which states that the precondition of a given exploit is never invalidated by the

Published by SK Research Group of Companies (SKRGC).

International Journal of Computer Science
Oddity...Probe...Reviste...
http://www.ijcsjournal.com Volume 2, Issue 2, No 1, 2014. ISSN: 2348-6600.
Reference ID: IJCS-054. PAGE NO: 311-319.

successful application of another exploit. In other words, attackers never need to backtrack.

With this assumption, they can obtain a concise, scalable graph representation for encoding attack tree. Ou et al. [12] proposed an attack graph tool called MulVAL, which adopts a logic programming approach and uses Datalog language to model and analyze network system. The attack graph in the MulVAL is constructed by accumulating true facts of the monitored network system. The attack graph construction process will terminate efficiently because the number of facts is polynomial in system. To provide the security assessment and alert correlation features, in this paper, we modified and extended MulVAL's attack graph structure. Intrusion Detection System (IDS) and firewall are widely used to monitor and detect suspicious events in the network.

However, the false alarms and the large volume of raw alerts from IDS are two major problems for any IDS implementations. To identify the source or target of the intrusion in the network, especially to detect multistep attack, the alert correction is a must-have tool. The primary goal of alert correlation is to provide system support for a global and condensed view of network attacks by analyzing raw alerts [13].

Many attack graph-based alert correlation techniques have been proposed recently. Wang et al. [14] devised an in memory structure, called queue graph (QG), to trace alerts matching each exploit in the attack graph. However, the implicit correlations in this design make it difficult to use the correlated alerts in the graph for analysis of similar attack scenarios. Roschke et al. [15] proposed a modified attack-graph-based correlation algorithm to create explicit correlations only by matching alerts to specific exploitation

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
Countermeasure Types

No.	Possible Countermeasure Types		
	Countermeasure	**Intrusiveness**	**Cost**
1	Traffic redirection	3	3
2	Deep Packet Inspection	3	3
3	MAC address change	2	1
4	Software patch	5	4
5	Network reconfiguration	0	5
6	Block port	4	1

A. Attack Analyser

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.

Cloud system information: collected from the node controller (Dom0 in Deserver). 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*

Published by SK Research Group of Companies (SKRGC).

International Journal of Computer Science
Oddity...Probe...Reviste...
http://www.ijcsjournal.com Volume 2, Issue 2, No 1, 2014. ISSN: 2348-6600.
Reference ID: IJCS-054. PAGE NO: 311-319.

configuration information: 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..*Virtual information:*generated by both on demand vulnerability scanning and regular penetration testing using the well-known vulnerability databases, such as Open Source Vulnerability Database (OSVDB) [25], Common Vulnerabilities and Exposures List (CVE) [23], and NIST National Vulnerability Database (NVD) [26].

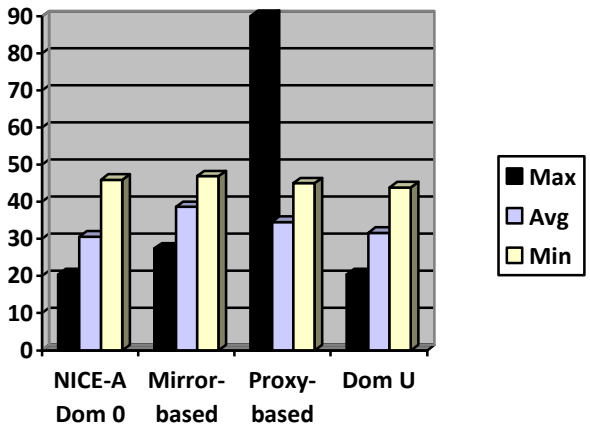

Fig.1. Network communication delay of NICE-A

B. *Private Cloud Environment*

Basically, vulnerability score considers the BSs of all the vulnerabilities on a VM. The BS depicts how easy it is for an attacker to exploit the vulnerability and how much damage it may incur. The exponential addition of BSs allows the vulnerability score to incline toward higher BS values and increases in logarithm-scale based on the number of vulnerabilities. The exploitability score on the other hand shows the accessibility of a target VM, and depends on the ratio of the number of services to the number of network services as defined in [33]. Higher exploitability score means that there are many possible paths for that attacker to reach the target.

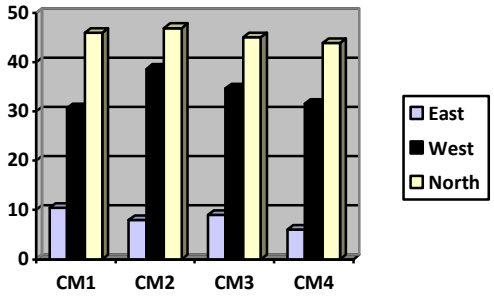

Fig. 1 Benefit evalution chart

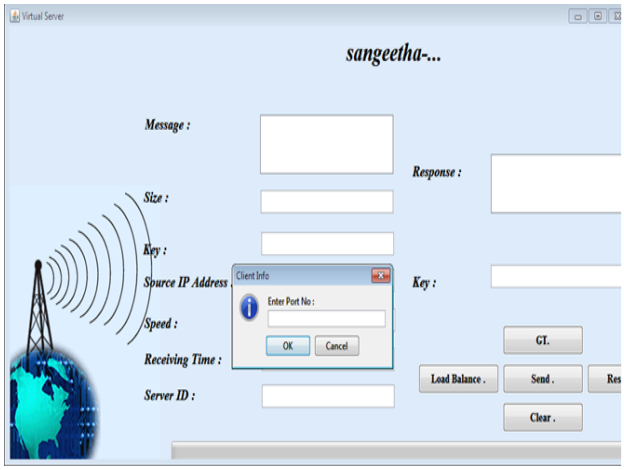

Fig. 2 Example of an image with acceptable resolution

Published by SK Research Group of Companies (SKRGC).

International Journal of Computer Science
Oddity...Probe...Reviste...
http://www.ijcsjournal.com Volume 2, Issue 2, No 1, 2014. ISSN: 2348-6600.
Reference ID: IJCS-054. PAGE NO: 311-319.

C. Security Performance Analysis

To demonstrate the security performance of NICE, created 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. 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 erver 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. the vulnerabilities present in this network and 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. 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 ¼ 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 of 85 percent, the probability of node 5 will change to 0.1164, ich causes change in probability values of all child nodes of node 5 thereby accumulating to a decrease of 28.5 percent 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) Apart from calculating the benefit measurements, we also present the evaluation based on ROI using (8) and represent a comprehensive evaluation considering benefit, cost, and intrusiveness of countermeasure. Fig. 6 shows the ROI evaluations for presented countermeasures. Results show that countermeasures CM2 and CM8 on node 5 have the maximum benefit evaluation; however, their cost and

Published by SK Research Group of Companies (SKRGC).

International Journal of Computer Science
Oddity...Probe...Reviste...
http://www.ijcsjournal.com Volume 2, Issue 2, No 1, 2014. ISSN: 2348-6600.
Reference ID: IJCS-054. PAGE NO: 311-319.

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.

Experiment in Private Cloud Environment

Previously, we presented an example where the attackers target is VM in the private network. For performance analysis and capacity test, we extended the configuration in Table 3 to create another test environment, which includes 14 VMs across three cloud servers and configured each VM

as a target node to create a dedicated SAG for each VM. These VMs consist of Windows (W) and Linux (L) machines in the private network 172.16.11.0/24, and contains more number of vulnerabilities related to their OSes and applications. We created penetration testing scripts with Metasploit framework [31] and Armitage [32] as attackers in our test environment. These scripts emulate attackers from different places in the internal and external sources, and launch diversity of attacks based on the vulnerabilities in each VM. To evaluate security level of a VM, we define a VSI to represent the security level of each VM in the current virtual network environment. This VSI refers to the VEA-bility metric [33] and utilizes two parameters that include Vulnerability and Exploitability as security metrics for a VM.

The VSI value ranges from 0 to 10, where lower value means better security. VSI: Definition 5 (VSI). VSI for a virtual machine k is defined as V SIk ¼ ðVk þ EkÞ=2, where 1. Vk is vulnerability score for VM k. The score is the exponential average of BS from each vulnerability in the VM or a maximum 10, i.e., Vk ¼ minf10; LnPeBaseScoreðvÞg. 2. Ek is exploitability score for VM k. It is the exponential average of exploitability score for all vulnerabilities or a maximum 10 multiplied by the ratio of network services on the VM, i.e., Ek ¼ ðminf10; lnPeExploitabilityScoreðvÞgÞ _ ð Sk NSk 3. Sk represents the number of services provided by VM k. NSk represents the number of network services the VM k can connect to.

Basically, vulnerability score considers the BSs of all the vulnerabilities on a VM. The BS depicts how easy it is for an attacker to exploit the vulnerability and how much damage it may incur. The exponential addition of BSs allows the vulnerability score to incline toward higher BS values and increases in logarithm-scale based on the number of vulnerabilities. The exploitability score on the other hand

shows the accessibility of a target VM, and depends on the ratio of the number of services to the number of network services as defined in [33]. Higher exploitability score means that there are many possible paths for that attacker to

reach the target. VSI can be used to measure the security level of each VM in the virtual network in the cloud system. It can be also used as an indicator to demonstrate the security status of a VM, i.e., a VMwith higher value of VSI means is easier to be attacked. To prevent attackers from exploiting other vulnerable VMs, the VMs with higher VSI values need to be monitored closely by the system (e.g., using DPI) and mitigation strategies may be needed to reduce the VSI value when necessary. plotting of VSI for these virtual machines before countermeasure selection and application. compares VSI values before and after applying the countermeasure CM4, i.e., creating filtering rules. It shows the percentage change in VSI after applying countermeasure on all of the VMs. Applying CM4 avoids vulnerabilities and causes VSI to drop without blocking normal services and ports.

Published by SK Research Group of Companies (SKRGC).

International Journal of Computer Science
Oddity...Probe...Reviste...
http://www.ijcsjournal.com Volume 2, Issue 2, No 1, 2014. ISSN: 2348-6600.
Reference ID: IJCS-054. PAGE NO: 311-319.

False Alarms

A cloud system with hundreds of nodes will have huge amount of alerts raised by Snort. Not all of these alerts can be relied upon, and an effective mechanism is needed to verify if such alerts need to be addressed. Since Snort can be programmed to generate alerts with CVE id, one approach that our work provides is to match if the alert is actually related to some vulnerability being exploited. If so, the existence of that vulnerability in SAG means that the alert is more likely to be a real attack. Thus, the false positive rate will be the joint probability of the correlated alerts, which will not increase the false positive rate compared to each individual false positive rate.

Moreover, we cannot keep aside the case of zero-day attack, where the vulnerability is discovered by the attacker but is not detected by vulnerability scanner. In such case, the alert being real will be regarded as false, given that there does not exist corresponding node in SAG. Thus, current research does not address how to reduce the false negative rate. It is important to note that vulnerability scanner should be able to detect most recent vulnerabilities and sync with the latest vulnerability database to reduce the chance of Zero-day attacks.

D. NICE System Performance

Evaluate system performance to provide guidance on how much traffic NICE can handle for one cloud server and use the evaluation metric to scale up to a large cloud system. In a real cloud system, traffic planning is needed to run NICE, which is beyond the scope of this paper. Due to the space limitation, we will investigate the research involving multiple cloud clusters in the future. To demonstrate the feasibility of our solution, comparative studies were conducted on several virtualization approaches. NICE based on Dom0 and DomU implementations with mirroring-based and proxybased attack detection agents (i.e., NICE-A). In mirrorbased IDS scenario, we established two virtual networks in each cloud server: Normal network and monitoring network.

NICE-A is connected to the monitoring network. Traffic on the normal network is mirrored to the monitoring network using Switched Port Analyzer (SPAN) approach. In the proxy-based IDS solution, NICE-A interfaces two VMs and the traffic goes through NICE-A. Additionally, we have deployed the NICE-A in Dom0 and it removes the traffic duplication function in mirroring and proxy-based solutions. NICE-A running in Dom0 is more efficient because it can sniff the traffic directly on the virtual bridge. However, in DomU, the traffic need to be duplicated on the VM's VIF, causing overhead. When the IDS is running in Intrusion Prevention System (IPS) mode, it needs to intercept all the traffic and perform packet checking, which consumes more system resources as compared to IDS mode. To demonstrate performance evaluations, Used four metrics namely CPU utilization, network capacity, agent processing capacity, and communication delay. Performed the evaluation on cloud servers with Intel quad-core Xeon 2.4-GHz CPU and 32-G memory. We used packet generator to mimic real traffic in the cloud system. As shown in Fig. 9, the traffic load, in the form of packet sending speed, increases from 1 to 3,000 packets per second. The performance at Dom0 consumes less CPU and the IPS mode consumes the maximum CPU resources.

It can be observed that when the packet rate reaches to 3,000 packets per second, the CPU utilization of IPS at DomU reaches its limitation, while the IDS mode at DomU only occupies about 68 percent. Represents the performance of

Published by SK Research Group of Companies (SKRGC).

International Journal of Computer Science
Oddity...Probe...Reviste...
http://www.ijcsjournal.com Volume 2, Issue 2, No 1, 2014. ISSN: 2348-6600.
Reference ID: IJCS-054. PAGE NO: 311-319.

NICE-A in terms of percentage of successfully analyzed packets, i.e., the number of the analyzed packets divided by the total number of packets received. The higher this value is, more packets this agent can handle. It can be observed from the result that IPS agent demonstrates 100 percent performance because every packet captured by the IPS is cached in the detection agent buffer. However, 100 percent success analyzing rate of IPS is at the cost of the analyzing delay.

For other two types of agents, the detection agent does not store the captured packets and, thus, no delay is introduced. However, they all experience packet drop when traffic load is huge. The communication delay with the system under different NICE-A is presented. We generated 100 consecutive normal packets with the speed of 1 packet per second to test the end-to-end delay of two VMs compared by using NICE-A running in mirroring and proxy modes in DomU and NICE running in Dom0. We record the minimal, average, and maximum communication delay in the comparative study. Results show that the delay of proxy-based NICE-A is the highest because every packet has to pass through it. Mirror-based NICE-A at DomU and NICE-A at Dom0 do not have noticeable differences in the delay. In summary, the NICE-A at Dom0 and Mirror-based NICE-A at DomU have better performance in terms of delay. From this test, we expected to prove the proposed solution, thus achieving our goal "establish dynamic defensive mechanism-based software defined networking approach that involves multiphase intrusion detections." The experiments prove that for a small-scale cloud system, our approach works well.

The performance evaluation includes two parts. First, security performance evaluation. It shows that the approach achieves the design security goals: To prevent vulnerable VMs from being compromised and to do so in less intrusive and cost effective manner. Second, CPU and throughput performance evaluation. It shows the limits of using the proposed solution in terms of networking throughputs based on software switches and CPU usage when running detection engines on Dom 0 and Dom U. The performance results provide us a benchmark for the given hardware setup and shows how much traffic can be handled by using a single detection domain. To scale up to a data center-level IDS, a decentralized approach must be devised, which is scheduled in our future research.

III. CONCLUSION

A novel technique for detecting application DOS attack by means of a new constraint-based group testing model. Motivated by classic GT methods, three detection algorithms were proposed and a system based on these algorithms was introduced. Theoretical analysis and preliminary simulation results demonstrated the outstanding performance of this system in terms of low detection latency and false positive/negative rate. Our focus of this paper is to apply group testing principles to application DOS attacks, and provide an underlying framework for the detection against a general case of network assaults, where malicious requests are indistinguishable from normal ones.

Future work will continue to investigate the potentials of this scheme and improve this proposed system to enhance the detection efficiency. Still improve it via false-tolerant group testing methods. This error-tolerant matrix has great potentials to improve the performance of the PND algorithm and handle application DOS attacks more efficiently.

Published by SK Research Group of Companies (SKRGC).

International Journal of Computer Science
Oddity...Probe...Reviste...
http://www.ijcsjournal.com Volume 2, Issue 2, No 1, 2014. ISSN: 2348-6600.
Reference ID: IJCS-054. PAGE NO: 311-319.

REFERENCES

1. 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.

2. 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.

3. 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.

4. N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic Security Risk Management Using Bayesian Attack Graphs," IEEE Trans.Dependable and Secure Computing, vol. 9, no. 1, pp. 61-74, Feb. 2012.

5. O. Database, "Open Source Vulnerability Database (OVSDB)," http://osvdb.org/, 2012.

6. R. Sadoddin and A. Ghorbani, "Alert Correlation Survey: Framework 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.

7. 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.

8. 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.

9. 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.

Published by SK Research Group of Companies (SKRGC).