Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 7, July 2015, pg.230 – 239

RESEARCH ARTICLE

EFFECTIVE APPROACH FOR CLOUD SECURITY USING NICE-D

Nikita T. Ramteke[1], Y.V. Chavhan[2]

[1]Department of Computer science, Savitribai Phule Pune University, India

[2]Department of Electronics, Savitaribai Phule Pune University, India

[1] nikita.ramteke306@gamil.com

[2] chavhan.yashwant@gmail.com

Abstract - Research has been evolved in the field of cloud security recently this few years. Where an attacker has the ability to explore various vulnerabilities of a system so that they can easily deploy further attacks. In network system including cloud especially in infrastructure-as-service (IaaS) cloud, the detection of attacks becomes very difficult. Therefore, existing system proposed multi-phase vulnerability detection measurement and countermeasure selection using NICE.

Existing NICE model uses signature based IDS i.e. SNORT [1]. In this paper, we improved the intrusion detection accuracy of NICE by using dynamic intrusion detection system (NICE-D). Dynamic IDS monitors the incoming traffic flow and anomalous time slot, and accordingly generates the new signature to identify the future intrusions over cloud system. The system and security evaluations demonstrate the efficiency and effectiveness of the proposed solution. Different attack scenarios have been considered for system evaluation.

Keywords – Iaas, SNORT, IDS, cloud, NICE-D.

I. Introduction:

Cloud computing is nothing but using a network consisting of remote servers which are hosted on internet to process the data, manage data and even store data. Cloud computing can also be termed as anything that is involved in delivering hosted services over the internet. Therefore there are number of security issues and concerns that are associated with cloud computing. These issues fall into two broad categories: first is security issues faced by cloud providers that is those organizations that provide software-as-a-service (SaaS), platform-as-a-service(PaaS), or infrastructure-as-a-service (IaaS) via the cloud and second is security issues faced by their customers. This paper aims for significantly prevent the virtual machines to become vulnerable from being compromised over the cloud server using multi-phase distributed vulnerability detection, measurement, and countermeasure selection mechanism called NICE-D.

In recent studies it have shown that, users those who are migrating to the cloud systems consider security as the most important factor. A recent Cloud Security Alliance (CSA) survey shows that among all security issues, abuse and notorious 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 [1]. 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 [3], which essentially contributes to loopholes in cloud security. Main task 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 [8] where cloud users have shared infrastructure, abuse and nefarious use of the shared infrastructure benefits attackers to exploit vulnerabilities of the cloud and also use its resource to deploy attacks in more easily and efficiently. As cloud users are sharing their computing resources it creates beneficial environment for attackers.

II. Proposed System:

In this paper, we propose NICE-D (Network Intrusion detection and Countermeasure selection in virtual network systems with Dynamic IDS) to establish a defense-in-depth intrusion detection framework. For better attack detection [2], NICE-D incorporates attack graph analytical procedures into the intrusion detection processes [2]. In NICE-D, we improve existing intrusion detection algorithms; NICE-D employs a reconfigurable virtual networking approach to detect and counter the attempts to compromise VMs, thus preventing zombie VMs.

The earlier contributions to NICE [1] are presented as follows:

1. We devised NICE-D [2], 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.
2. NICE can improve the attack detection probability and improve the resiliency to VM exploitation attack without interrupting existing normal cloud services [1].
3. NICE employs a novel attack graph approach for attack detection and prevention by correlating attack behavior and also suggests effective countermeasures [1].
4. 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.

Advanced Contributions:

1. Use of Dynamic intrusion detection system [2].
2. Proposed system will be able to improve the false alarm rate [2].
3. Proposed system will detect the future intrusions in operational environment.

III. System Architecture

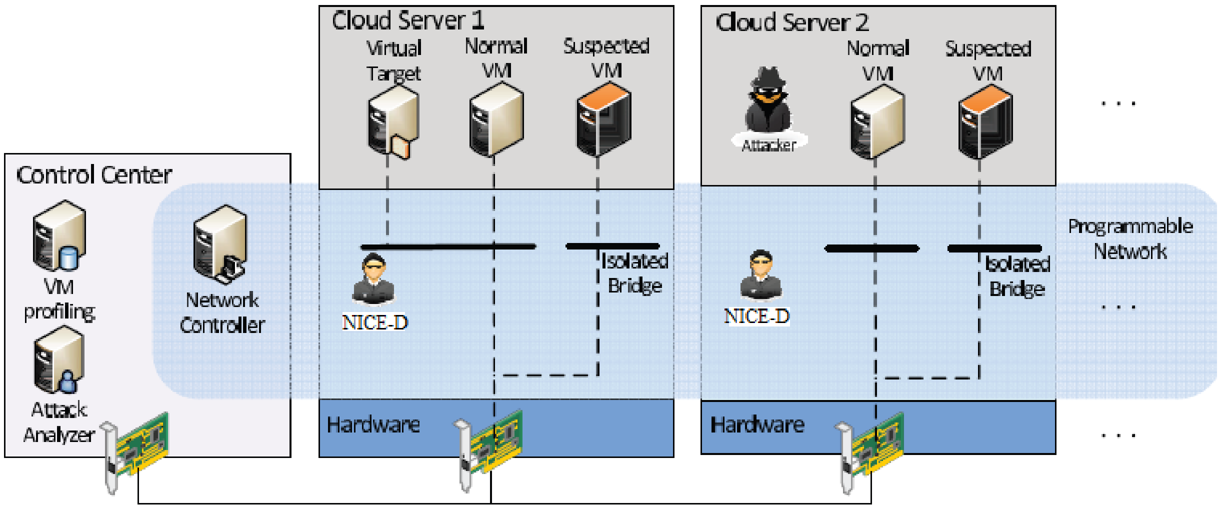


Fig1. System architecture

Intrusion detection systems are commonly used by network administrators to monitor the traffic being exchanged between different network segments. And by replacing the traditional local server-based network environments with cloud-based network infrastructure, system administrators will need to purchase additional services from the cloud provider so that they can deploy their own network intrusion detection systems. For this reason NICE-D is incorporated for cloud security. Fig 1. Shows system architecture where there is Control Center consisting of VM profiling, Attack Analyzer and Network Controller. Cloud Servers are associated with Cloud centre and in each cloud server we have NICE-D agent that will provide security. The main discussion is mainly about the effective design of an intrusion detection system that can be integrated with the available services in cloud networks. The main idea is to provide intrusion detection as a service for the cloud users. The required intrusion detection framework has some desired criteria, where we have NICE-D at every cloud server associated with the control centre.

IV. Algorithm Used:

A. Dynamic IDS:

Step 1: Start
Step 2: capture traffic ,packet
Step 3: built time series
Step 4: Identify anomalous time slot
Step 5: treat suspicious flow as evidence
Step 6: Production of filtering rule
Step 7: formation of signature
Step 8: detect intrusion
Step 9: Generate Alarm
Step 10: Stop

In first stage using a temporal sliding-window approach, traffic is captured and then aggregated in flows. This is done using different levels of traffic aggregation. Simple traffic metrics such as number of bytes flows in each time slot, time series are built.

In the second stage unsupervised detection algorithm begins. Output of previous stage is used in this stage as a input. Filtering rules helps to provide characteristic of attack [8]. But when the network operator deal with the unknown attack, the characterization of attack may became much more difficult as it require good, easy, simple information as the input [8]. To deal with this issue, new traffic signature is developed by combining relevant filtering rules. Developed signature will detect the attack coming in future. This is the important step toward autonomous security.

B. Alert Correlation Algorithm

```
Algorithm 1 Alert_Correlation
Require: alert $a_c$, $SAG$, $ACG$
1: if ($a_c$ is a new alert) then
2:     create node $a_c$ in $ACG$
3:     $n_1 \leftarrow v_c \in map(a_c)$
4:     for all $n_2 \in parent(n_1)$ do
5:         create edge $(n_2.alert, a_c)$
6:         for all $S_i$ containing $a$ do
7:             if $a$ is the last element in $S_i$ then
8:                 append $a_c$ to $S_i$
9:             else
10:                create path $S_{i+1} = \{subset(S_i, a), a_c\}$
11:            end if
12:        end for
13:        add $a_c$ to $n_1.alert$
14:    end for
15: end if
16: return $S$
```

Attack graph is a modeling tool to illustrate possible multihost, multistage attack paths that are important to understand threats and then to decide appropriate countermeasure [2]. In an attack graph, each node represents either precondition or consequence of an exploit [2]. Attack graphs are used to provide information about the current security situation in the system.

If any event is recognized as potential treat then we can apply specific countermeasure to mitigate the impact of that attack.

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. [12] and define as Scenario Attack Graph (SAG) [1]. We consider the definition of SAG as explained in the [1].

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 [1]. To keep track of attack status, we track the source and destination IP addresses for attack activities [1].

Definition of alert correlation graph is used from [1].

Alert correlation algorithm is used for every alert detected and gives one or more paths Si. Every alert ac that is received from the detection system is added to ACG if it does not exist. This new alert is mapped with corresponding vertex in the SAG using function map (ac). For this vertex in SAG, alert related to its parent vertex of type NC is then correlated with the current alert *ac* which creates a new set of alerts that belong to a path Si in ACG.

C. Countermeasure Selection Algorithm

Algorithm 2 Countermeasure_Selection

```
Require: $Alert, G(E,V), CM$
1:  Let $v_{Alert}$ = Source node of the $Alert$
2:  if Distance_to_Target($v_{Alert}$) $> threshold$ then
3:      Update_ACG
4:      return
5:  end if
6:  Let $T = Descendant(v_{Alert}) \cup v_{Alert}$
7:  Set $Pr(v_{Alert}) = 1$
8:  Calculate_Risk_Prob($T$)
9:  Let $benefit[|T|, |CM|] = \emptyset$
10: for each $t \in T$ do
11:     for each $cm \in CM$ do
12:         if $cm.condition(t)$ then
13:             $Pr(t) = Pr(t) * (1 - cm.effectiveness)$
14:             Calculate_Risk_Prob($Descendant(t)$)
15:
                $$benefit[t, cm] = \Delta Pr(target\_node). \quad (7)$$
16:         end if
17:     end for
18: end for
19: Let $ROI[|T|, |CM|] = \emptyset$
20: for each $t \in T$ do
21:     for each $cm \in CM$ do
22:
            $$ROI[t, cm] = \frac{benefit[t, cm]}{cost.cm + intrusiveness.cm}. \quad (8)$$
23:     end for
24: end for
25: Update_SAG and Update_ACG
26: return Select_Optimal_CM($ROI$)
```

Counter measure selection algorithm presents the how to select optimal countermeasure for given attack scenario. This algorithm uses alert, attack graph and pool of countermeasure as input. An algorithm starts by selecting the node vAlert that corresponds to the alert generated by a NICE-D [1]. Countermeasure selection is done by counting 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 [1]. For the source node vAlert, all the reachable nodes are collected into a set T [1]. Alert is generated only after the attacker has performed the action, The probability of vAlert is set to 1 and calculated the new probabilities for all of its child nodes in the set T [1].

For all alerts t ϵ T the applicable countermeasures in CM are selected and new probabilities are calculated according to the effectiveness of the selected countermeasures [2]. The change in probability of target node gives the benefit for the applied countermeasure [1]. We compute the Return of Investment (ROI) for each benefit of the applied countermeasure based on. 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 [1].

V. NICE System Component

There are basically four major components that will be included in our system. They are,

- Nice-D
- VM Profiling
- Attack Analyzer
- Network Controller

Nice-D

The NICE-D is a Network-based Intrusion Detection System (NIDS) with dynamic IDS installed in each cloud server [2]. It scans the traffic going through the bridges that control all the traffic among VMs and in/out from the physical cloud servers [2]. 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 and ERSPAN methods. This method is efficient to scan the traffic in cloud server as all traffic in the cloud server needs go through it.

VM Profiling

Virtual machines in the cloud can be profiled to get important information about their state, services running, open ports [1].Important factor in VM profile is its connectivity with other VMs [1]. Knowledge of services running on a VM is necessary to verify the authenticity of alerts pertaining to that VM. Information about any open ports on a VM and the history of opened ports plays important role in determining how vulnerable the VM is.

Attack Analyzer

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

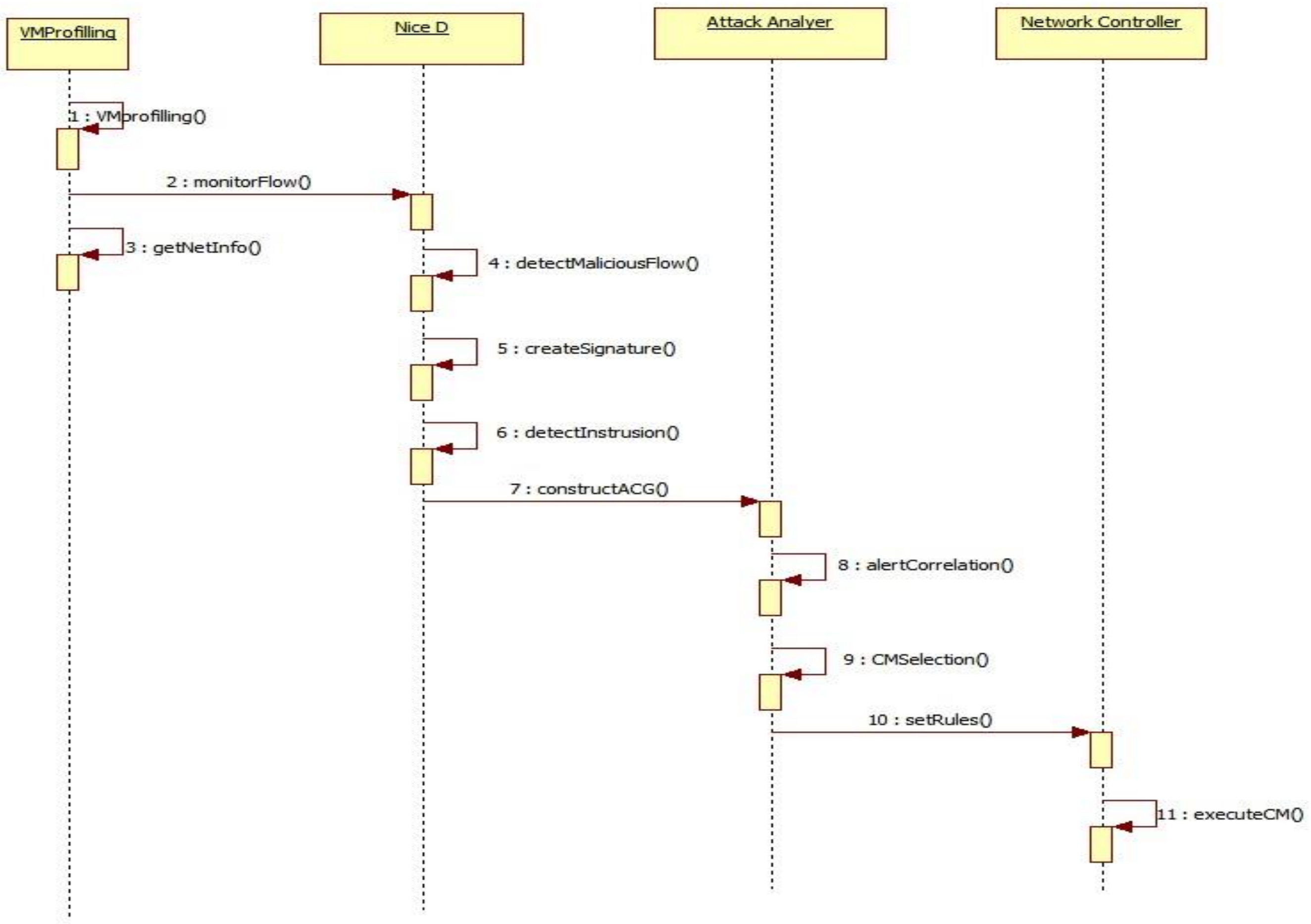
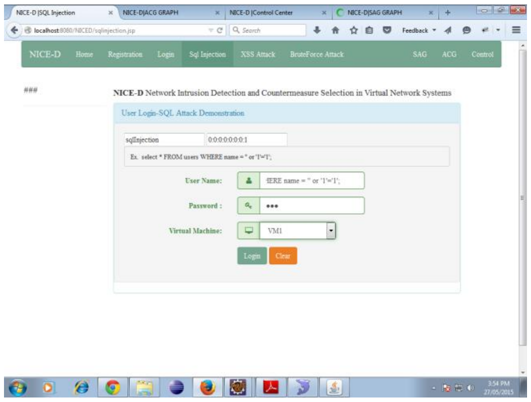


Fig 2. Sequence diagram for Nice system Components

Network Controller

The network controller is a key component to support the programmable networking capability to realize the virtual network reconfiguration [2]. In NICE, we integrated the control functions for both OVS and OFS into the network controller which permits the cloud system to set security/filtering rules in an integrated and comprehensive manner [1]. The network controller is responsible for collecting network information of current Open Flow network, which provides input to the attack analyzer to construct attack graphs. Network controller 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.

VI. Results and Discussion


NICE-D Network Intrusion Detection and Countermeasure Selection in Virtual Network Systems

User Login-SQL Attack Demonstration

sqlInjection 0:0:0:0:0:0:0:1

Ex. select * FROM users WHERE name = '' or '1'='1';

User Name: IERE name = '' or '1'='1';

Password : •••

Virtual Machine: VM1

Fig 3. SQL attack

Above figure shows the attack demonstration for sql attack. Here different sql attacks have been performed using sql keywords where attacker can try to perform attack using sql queries and in that case our system will capture those attacks and will block the respective ip address. Our system shows the effectiveness by detecting such type of attack over cloud server. Also XSS attack, Login attempt attack have been demonstrated.

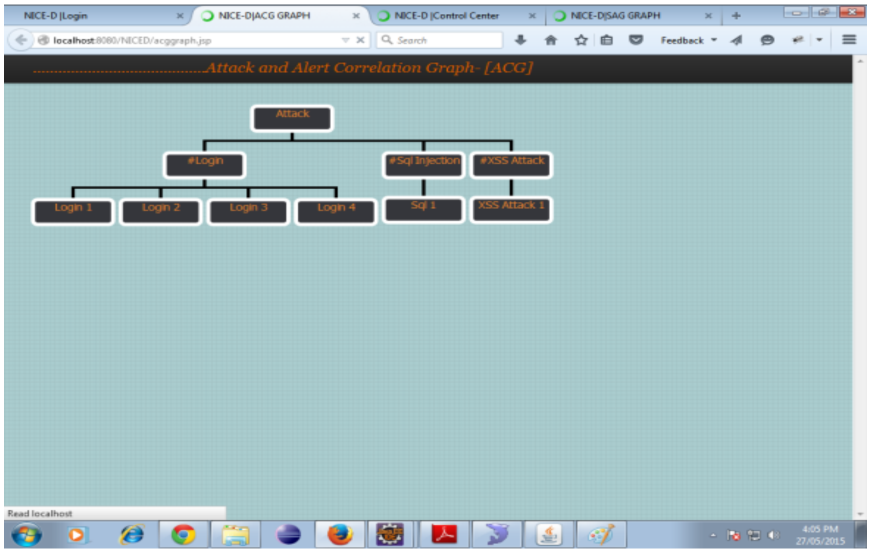


Fig 4 . ACG graph

Attack co-relation graph (ACG), shows the different attack that have been used to attack a cloud system. This graph shows the different nodes and path for different attacks from the root node. Both the graphs that is ACG and SAG comes under attack analyzer whose main function is to plat the graphs for further observations.

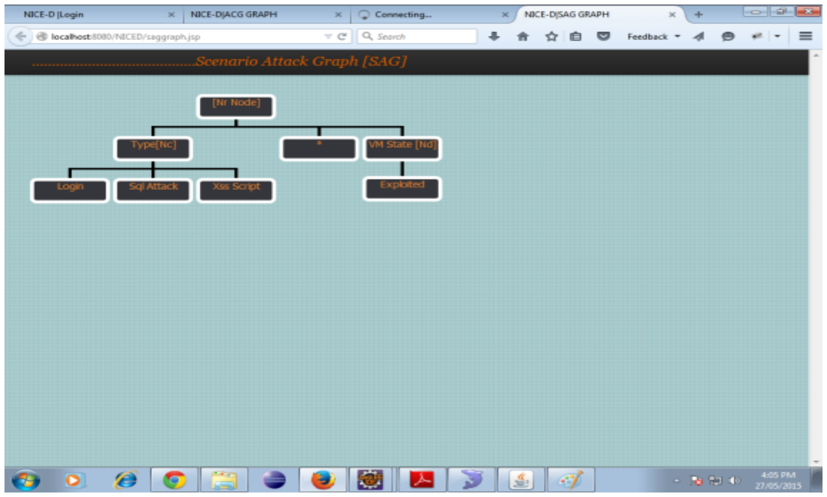
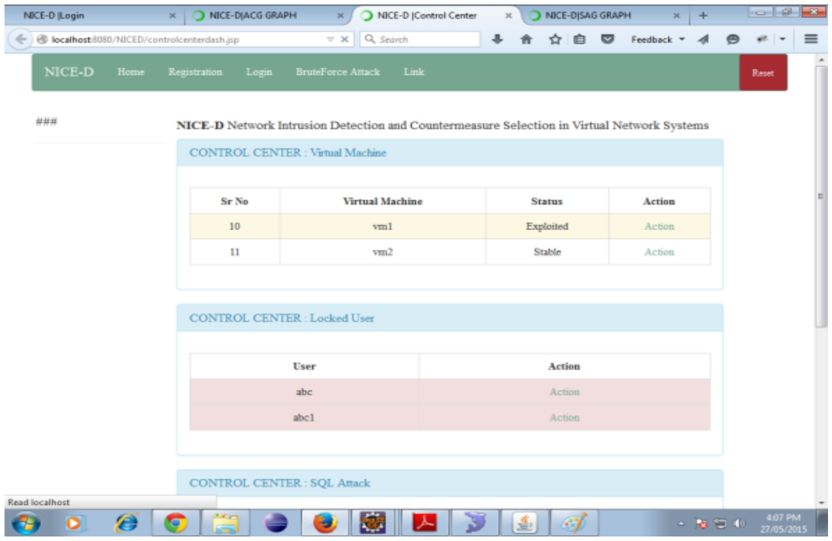


Fig 5 . SAG graph



NICE-D Home Registration Login BruteForce Attack Link Reset

###

NICE-D Network Intrusion Detection and Countermeasure Selection in Virtual Network Systems

CONTROL CENTER : Virtual Machine

Sr No	Virtual Machine	Status	Action
10	vm1	Exploited	Action
11	vm2	Stable	Action

CONTROL CENTER : Locked User

User	Action
abc	Action
abc1	Action

CONTROL CENTER : SQL Attack

Fig 6. Network Controller

Above figure shows that there is control where controller can observe the status and actions to be taken on attacks and here the controller can reset all the values also.

Conclusion:

NICE-D for cloud system work better than the existing system NICE-A. NICE-A covered all possible work for the VM over cloud but did not considered the improvement to the intrusion detection algorithm. In this paper we implemented and demonstrated the new dynamic intrusion detection algorithm which performs better than the existing signature based SNORT. We have discussed the implementation results of proposed system.

References:

[1] Chun-Jen Chung, Student Member, IEEE, Pankaj Khatkar, Student Member, IEEE, Tianyi Xing, Jeongkeun Lee, Member, IEEE, and Dijiang Huang Senior Member, IEEE-"NICE: Network Intrusion Detection and Countermeasure Selection in Virtual Network Systems"- IEEE TRANSACTIONS ON DEPEDABLE AND SECURE COMPUTING, 2013.

[2] Nikita, Y. chavan "NICE-D: A MODIFIED APPROACH FOR CLOUD SECURITY" IJSR volume 2 issue 12, 2014.

[3] Coud Sercurity Alliance, "Top Threats to Cloud Computing v1.0," https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf, Mar. 2010.

[4] 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.

[5] 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.

[6] 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.

[7] "Open vSwitch Project," http://openvswitch.org, May 2012.

[8] 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.

[9] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "BotHunter: Detecting Malware Infection through IDS-driven Dialog Correlation," Proc. 16th USENIX Security Symp. (SS '07), pp. 12:1-12:16, Aug. 2007.

[10] 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.

[11] Michael Armbrust, Armando Fox, Rean Griffith "Above the Clouds: A View of Cloud Computing" Tech. Rep. UCB/EECS-2009-28, EECS Department, U.C. Berkeley, Feb 2012.

[12] Ms. Parag K. Shelke, Ms. Sneha Sontakke, Dr. A. D. Gawande "Intrusion Detection System for Cloud Computing "International Journal of Scientific & Technology Research Volume 1, Issue 4, May 2012.

[13] J. Sasi Devi, R. Sugumar "Host Based Intrusion Detection to Prevent VirtualNetwork System from Intruders in Cloud" International Journal of Science and Research (IJSR) 2014