RESEARCH ARTICLE OPEN ACCESS

Detecting Malware Intrusion in Network Environment

Mr. V.V.Prathap[1], Mrs.D.Saveetha[2]
[1, 2] SRM University

ABSTRACT
Over a Past Few YearsCloud security is one of most important issues that has attracted a lot of research and development,Especially Attackers can Find explore vulnerabilities of a cloud system and compromise virtual machines to deploy further large-scale Distributed Denial-of-Service (DDoS).As DDoS Attacks Usually Involves Early Stage Actions the Detection of Zombie Exploration Attacks is Extremely Difficult Because of Cloud Users May Install Vulnerable Applications on Their Virtual Machines .To Prevent this Condition we Propose a Multi-Phase ,Distributed Vulnerability Detection Measurement and Counter Measure Selection Mechanism called NICE Implementation. This Model is built on Attack Graph Based Analytical Models and Re-Configurable Virtual Network Based Counter Measures. A Scenario Attack Graph Technique is Used to Prevent the Attacker while he wants to Enter to Other User/Server in the Network. As NICE Proposes Three Models, Where Scenario Attack Graph is Proved asPreferred Model.

***INDEX TERMS*:** Attack Graph, Cloud Computing, Intrusion Detection, Network Security, Zombie Detection

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 centres, 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 centres, where cloud users usually have the privilege to control softwareinstalled 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.

To establish a defence-in-depth Intrusion Detection Framework, We Propose NICE. 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

Actually, NICE includes two main phases: (1) deploy a lightweight mirroring based network intrusion detection agent (NICE-A) on each cloud server to capture and analyse cloud traffic. A NICE-A periodically scans the virtual systemvulnerabilities within a cloud server to establish Scenario Attack Graph (SAGs), and then based on the severity of identified vulnerability towards the collaborative attack goals, NICE 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 behaviours prominent.

II. NICE MODELS

Basically,NICE Consists of Three Models

2.1 Threat Model

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 Infrastructure-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 vulnerability.

2.2 Attack Graph Model

An attack graph is a modelling tool to illustrate all possible multi-stage, multi-host attack paths that are crucial to understand threats and then to decide appropriate countermeasures.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.

Definition 1(Scenario Attack Graph)**.** *An Scenario Attack Graph is a tuple SAG=(V, E), where*
1. *$V = NC \cup ND \cup NR$ denotes a set of vertices that include three types namely conjunction node NC to represent exploit, disjunction node ND to denote result of exploit, and root node NR for showing initial step of an attack scenario.*

2. *$E = Epre \cup Epost$ denotes the set of directed edges. An edge $e \in Epre \subseteq ND \times NC$ represents that NDmust be satisfied to achieve NC. An edge $e \in Epost \subseteq NC \times ND$ means that the consequence shown by NDcan be obtained if NC is satisfied.* Node *$vc \in NC$* is defined as a three tuple(*Hosts, vul, alert*) representing a set of IP addresses,vulnerability information such as CVE [23], and alertsrelated to *vc*, respectively. *ND* behaves like a logical ORoperation and contains details of the results of actions.*NR* represents the root node of the scenario attack graph.

Definition 2(Alert Correlation Graph)**.** *An ACG is a three tuple ACG = (A,E, P), where*
1. *A is a set of aggregated alerts. An alert $a \in A$ is a data structure (src, dst, cls, ts) representing source IP address, destination IP address, type of the alert, and timestamp of the alert respectively.*

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

3. *E is a set of directed edges representing correlation between two alerts (a, a_) if criteria below satisfied:*
i. $(a.ts < a\_.ts) \wedge (a\_.ts - a.ts < threshold)$
ii. $\exists (vd, vc) \in Epre : (a.dst \in vd.Hosts \wedge a\_.src \in vc.Hosts)$

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 ratherthan raw alerts. Raw alerts having same source anddestination IP addresses, attack type and timestampwithin a specified window are aggregated as *Meta Alerts*.

III. Related Theory

3.1 Existing Model

In the Existing System, When an Attacker Attacks the User/Server in the Network which are especially Infrastructure-as-a-Service[IaaS] based Servers, detection of effected Servers are Extremely Difficult because of cloud Users may install multiple Types of Software in the Server with their User Account.Existing work generally focuses on measuring individual vulnerabilities instead of measuring their combined effects.

3.2 Proposed Model

In the Proposed System, We propose Network Intrusion detection and Countermeasure Selection to establish a defense-in-depth intrusion detection framework for better attack detection, Network Intrusion detection and Countermeasure Selection incorporates attack graph analytical procedures into the intrusion detection processes.

When an Attacker Attacks the Server by using a User Account, Attacker can Deploy Multiple Levels of Malwares to the Server, If and only if he can Access to the Server, but in Existing System its Hard to Detect the Attacker because of Server Cloud Service . While in Proposed, When an Attacker Attacks the Server using User Account, the Attack Analyzer can Detect the Attacker and Send the Warning to Administrator that User[Attacked by the Zombie] try to Access to Other Users Account to Deploy the Multiple Levels of Malware and Admin waits for Maximum Attempts and then Admin Blocks him Permanently using Scenario Attack Graph.

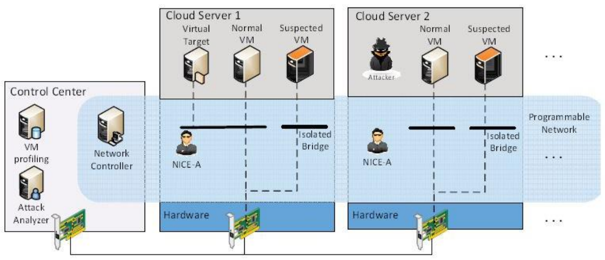


Fig 1 : Designed NICE Architecture

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 model 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.

Algorithm ::
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
```

Above method for utilizing SAG and ACG together so as to predict an attacker's behaviour. Alert Correlation algorithm is followed for every alert detected and returns one or more paths Si.For every alert ac that is received from the IDS, it is added to ACG if it does not exist. For thisnew alert ac, the corresponding vertex in the SAG is found by using function map.

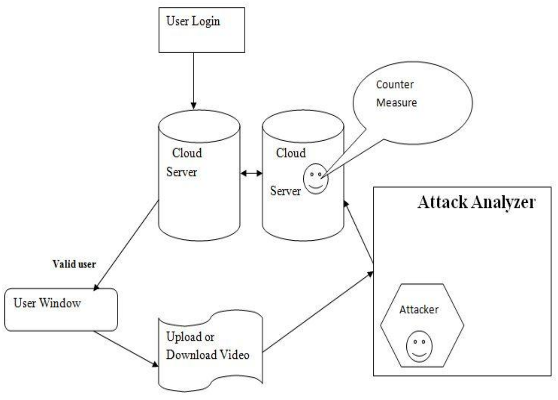


Fig 2 : Counter-Measure Model

Algorithm 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 electing the node vAlert that corresponds to the alert generated by a NICE-A. Before selecting the countermeasure, we count 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.

IV. Step-by-Step Procedure to Prevent Attack

The below Figures Shows Every Moment of Application while Running It gives the clear elaborated of application. It will be useful for the new user to understand for the future steps.

Intrusion Detection Model:

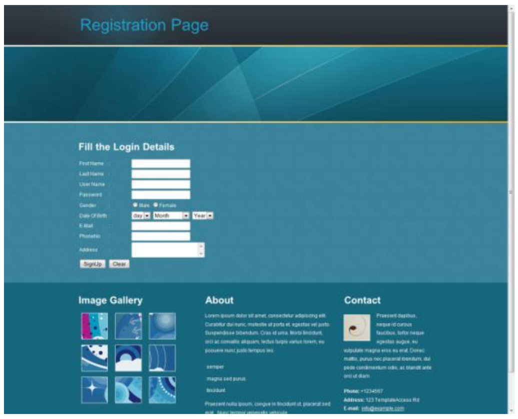


Fig3 : Registration Process

In above figure describes registration process where user is provides his own details for registering and he is will be getting login on successful registration or else if he fails to provide any of the details he will not be allowed to register thereby he is not allowed to login

Attack Graph Model:

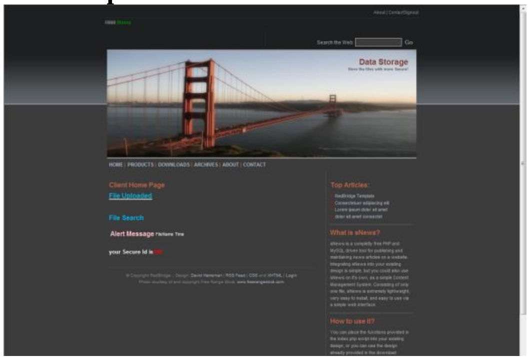


Fig 4 : User Profile

After successful login he is allowed to enter the application profile, from there onwards what are the information he may want to get he is simply access from the application

Upload Information

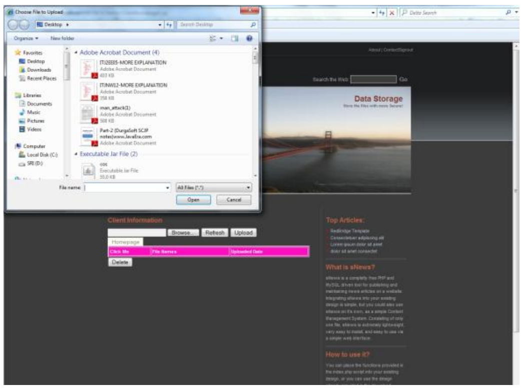
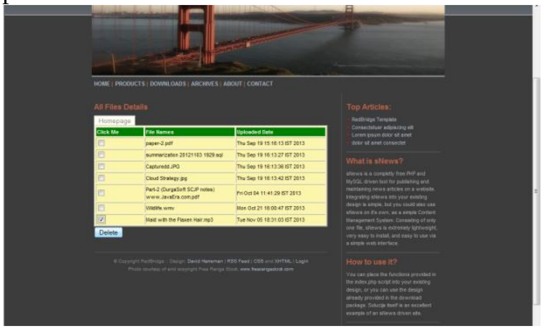


Fig 5 : Uploading a File After Login

In Above Figure describes that a Registered User can Access his own Account by Uploading any type of files, he can only able to have the delete Option .


All Files Details

Homepage

Click Me	File Names	Uploaded Date
☐	paper-2.pdf	Thu Sep 19 15:16:13 IST 2013
☐	summarization 20121103 1929.sql	Thu Sep 19 16:13:27 IST 2013
☐	Capturedd.JPG	Thu Sep 19 16:13:36 IST 2013
☐	Cloud Strategy.jpg	Thu Sep 19 16:13:42 IST 2013
☐	Part-2 (DurgaSoft SCJP notes) www.JavaEra.com.pdf	Fri Oct 04 11:41:29 IST 2013
☐	Wildlife.wmv	Mon Oct 21 16:00:47 IST 2013
✓	Maid with the Flaxen Hair.mp3	Tue Nov 05 18:31:03 IST 2013

Delete

Fig 6 : All Files Option

In Above Figure describes the User can check the All Files Available in his Cloud Server by clicking on “All files” Option in his Account

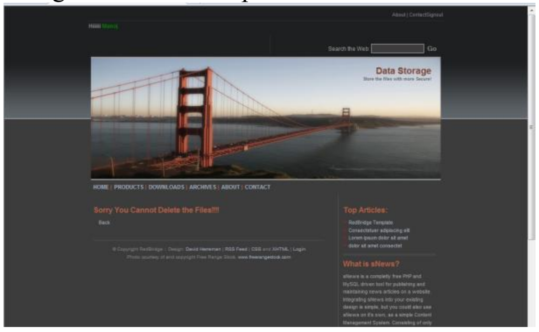


Fig 7 :Preventing the Deletion

In Above Figure describes, if User tries to Delete a file which is Available in Cloud Server it shows *“Sorry You Cannot Delete the Files!!!!”*because the User will having only “Read-Write-Remove[RWR]” only to his Account not to Remaining Account/Files

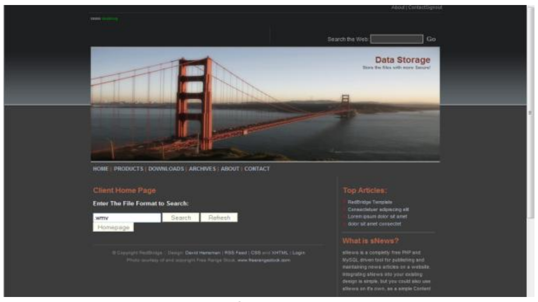


Fig 8 :: Selection of a Particular File/Format

In Above Figure describes if User having ‘n’ number of files and he wants to check a particular file/Format, he can access that File in the Search Option by Typing the File name/Format Type he wants

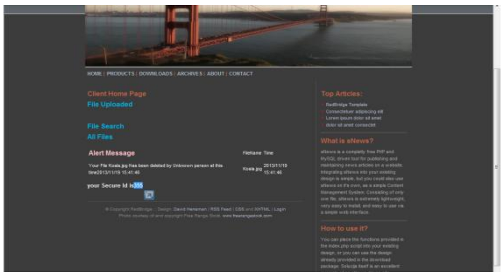


Fig 9 : Providing a Unique ID

In Above Figure describes that for every login User having an Unique Secure ID in the Application, for the Security Reasons and ID is useful when User request to Administrator to Delete a File, User wants to Submit Secure ID with the Message which is showing in above figure

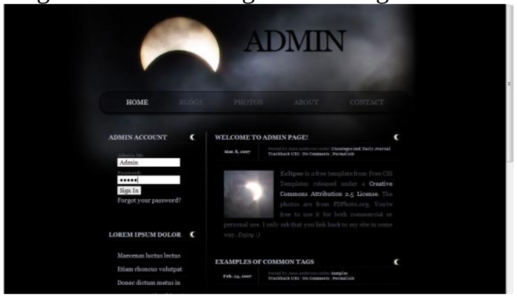


Fig 10 : Admin Login Page

In Above Figure describes the Administrator Login Page

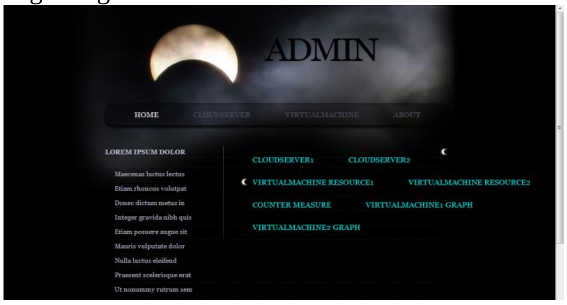
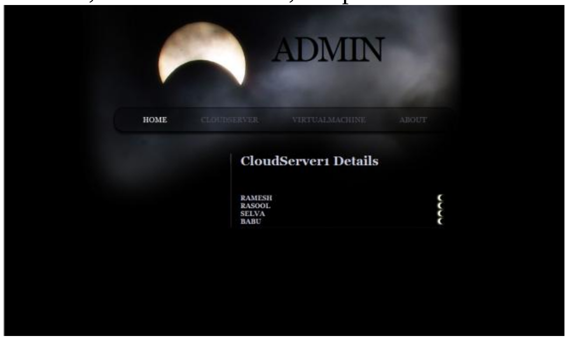


Fig 11 :Admin Index

After Login, Administrator Having an 7 Options Index of his Cloud Server and Virtual Machine, Counter-Measure, Graph

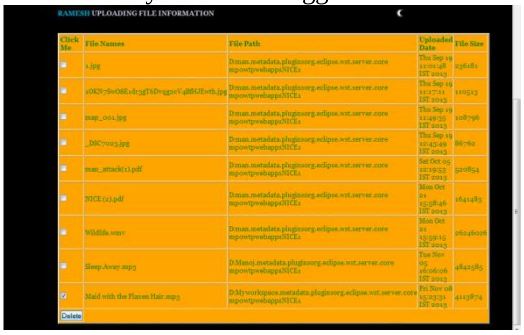


CloudServer1 Details

RAMESH
RASOOL
SELVA
BABU

Fig 12 : Accessing the Details

In Above Figure describes the Administrator, Access to Cloudserver1 Details which shows how many Users are Logged in



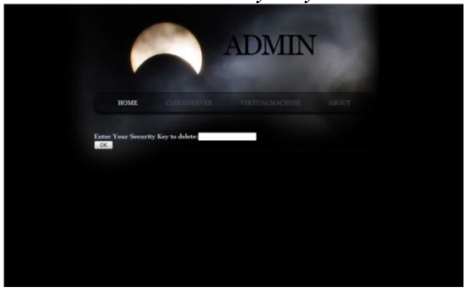
RAMESH UPLOADING FILE INFORMATION

Click Me	File Names	File Path	Uploaded Date	File Size
☐	1.jpg	D:man.metadata.pluginsorg.eclipse.wst.server.core mpowtpwebappsNICE1	Thu Sep 19 11:01:48 IST 2013	236181
☐	1OKN-7SwO8E1dr3gT6Dvqg2cV4B8HJEwth.jpg	D:man.metadata.pluginsorg.eclipse.wst.server.core mpowtpwebappsNICE1	Thu Sep 19 11:17:11 IST 2013	110513
☐	map_001.jpg	D:man.metadata.pluginsorg.eclipse.wst.server.core mpowtpwebappsNICE1	Thu Sep 19 11:49:35 IST 2013	108796
☐	_DSC7023.jpg	D:man.metadata.pluginsorg.eclipse.wst.server.core mpowtpwebappsNICE1	Thu Sep 19 12:43:49 IST 2013	86760
☐	man_attack(1).pdf	D:man.metadata.pluginsorg.eclipse.wst.server.core mpowtpwebappsNICE1	Sat Oct 05 12:19:53 IST 2013	520854
☐	NICE (2).pdf	D:man.metadata.pluginsorg.eclipse.wst.server.core mpowtpwebappsNICE1	Mon Oct 21 15:58:46 IST 2013	1641483
☐	Wildlife.wmv	D:man.metadata.pluginsorg.eclipse.wst.server.core mpowtpwebappsNICE1	Mon Oct 21 15:59:15 IST 2013	26246026
☐	Sleep Away.mp3	D:Manoj.metadata.pluginsorg.eclipse.wst.server.core mpowtpwebappsNICE1	Tue Nov 05 16:06:06 IST 2013	4842585
☑	Maid with the Flaxen Hair.mp3	D:Myworkspace.metadata.pluginsorg.eclipse.wst.server.core mpowtpwebappsNICE1	Fri Nov 08 15:23:31 IST 2013	4113874

Delete

Fig 13 : Cloud Server Files

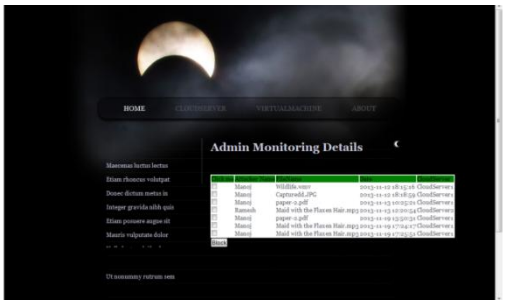
In Above Figure describes the Administrator, having an Access to the Particular Users data in the Cloud server, if Admin tries to delete a File in the User Account, he can't able to delete a file without "*Security Key"*



Enter Your Security Key to delete:
OK

Fig 14 : Security Key Request

In Above Figure describes the Administrator, tries to delete a File in the User Account only by Entering "Security Key" which is generated when User Login ,which is shown in Fig 9



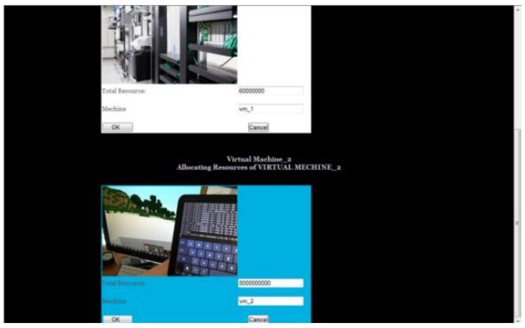
Admin Monitoring Details

Click me	Attacker Name	fileName	Date	CloudServer
☐	Manoj	Wildlife.wmv	2013-11-12 18:15:16	CloudServer1
☐	Manoj	Capturedd.JPG	2013-11-12 18:18:59	CloudServer1
☐	Manoj	paper-2.pdf	2013-11-13 10:25:21	CloudServer1
☐	Ramesh	Maid with the Flaxen Hair.mp3	2013-11-13 12:20:54	CloudServer2
☐	Manoj	paper-2.pdf	2013-11-19 13:50:31	CloudServer1
☐	Manoj	Maid with the Flaxen Hair.mp3	2013-11-19 17:24:17	CloudServer1
☐	Manoj	Maid with the Flaxen Hair.mp3	2013-11-19 17:25:51	CloudServer1

Block

Fig 15 :Admin Monitoring Details

In Above Figure describes the Administrator, Access to Counter-Measurewhich is in that a User1[Attacker] who tries to delete files of other User2, Admin can see the how many Attempts that Attacker made to delete Files of other User. Admin can also Block the Attacked User who tries to delete other User



Total Resource: 60000000
Machine: vm_1
OK Cancel

Virtual Machine_2
Allocating Resources of VIRTUAL MACHINE_2

Total Resource: 6000000000
Machine: vm_2
OK Cancel

Fig 16 :Access to VM

In Above Figure describes the Administrator Access to Virtual Machine Allocating Storage date of Virtual Machine

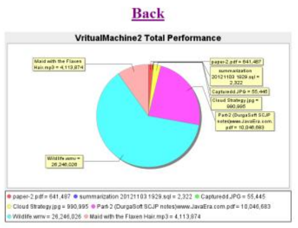


Fig 17: VM Graph

In Above Figure describes the Administrator having Access to Virtual Machine Graph which shows Data Usage of Particular Virtual Machine in Cloud Server

V. 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 indicatedin 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] Cloud Security Alliance “Top threats to cloud computing 1.0, 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.

[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/∼nusmv. 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.