International Journal of Computer Applications Technology and Research
Volume 3– Issue 11, 737 - 744, 2014, ISSN: 2319–8656

Evasion Streamline Intruders Using Graph Based Attacker model Analysis and Counter measures In Cloud Environment

D.Usha Sree
Chiranjeevi Reddy Institute of Technology
Anantapuramu-51500,
Andhra Pradesh. India

S. Sravani
Chiranjeevi Reddy Institute of Technology
Anantapuramu-51500,
Andhra Pradesh.India

Abstract: Network Intrusion detection and Countermeasure Election in virtual network systems (NICE) are used 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. NICE includes two main phases: deploy a lightweight mirroring-based network intrusion detection agent (NICE-A) on each cloud server to capture and analyze cloud traffic. A NICE-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, NICE will decide whether or not to put a VM in network inspection state. 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.

Keywords: NICE, Compromised Machines, spam zombies, Compromised Machine detection Algorithms Scenario Attack Grapg(SAGs)

1. INTRODUCTION

RECENT studies have shown that users migrating to the cloud consider security as the most important factor. A recent Cloud Security Alliance (CSA) [1] .Survey shows that among all security issues, abuse and nefarious use of cloud computing [2] 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 [3] 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 [4]. 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. Addressed that protecting "Business continuity and services availability" from service outages is one of the top concerns in cloud computing systems.

1.1 Motivation

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 bump-in-the-wire (i.e., proxy-based) 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. Based on the collective behavior of VMs in the SAG, NICE can decide appropriate actions, for example, DPI or traffic filtering, on the suspicious VMs. Using this approach, NICE does not need to block traffic flows of a suspicious VM in its early attack stage.

1.2 Definitions

NICE is a new multiphase distributed network intrusion detection and prevention framework in a virtual networking environment that captures and inspects suspicious cloud traffic without interrupting users' applications and cloud services.

NICE incorporates a software switching solution to quarantine and inspect suspicious VMs for further investigation and protection. Through programmable network approaches, NICE can improve the attack detection probability and improve the resiliency to VM exploitation attack without interrupting existing normal cloud services.

NICE employs a novel attack graph approach for attack detection and prevention by correlating attack behavior and also suggests effective countermeasures.

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

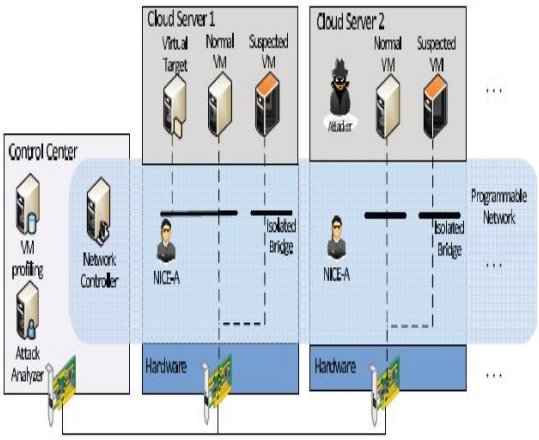

Figure 1.1: Architecture of intruders

2. PROBLEM STATEMENT

2.1 Existing System

Every day in data repositories many number of knowledgeable people are update the data. Data is increases here. Already existing data it can add in two or more number of databases. These kinds of data repositories are come under dirty repositories. Any user it can forward the query, extract and display the results here. Extraction results are contains useless data. Query shows much number of problems like high response amount of time, availability, quality assurance and security. Websites are not providing any useful services in extraction. These services are showing the problems in performance, quality and operational cost.

Previous existing system applies the data integration, data cleaning under record linkage and record matching. In record matching time and record linkage any duplicates are present removed here. Next previous approach near duplicate detection also remove some duplicates of data. These approaches are not gives any efficient solution in implementation. It cannot provide high quality data.

2.2 Proposed System

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 loopholes in cloud security.

In a cloud system, where the infrastructure is shared by potentially millions of users, abuse and nefarious use of the shared infrastructure benefits attackers to exploit vulnerabilities of the cloud and use its resource to deploy attacks [5] in more efficient ways. 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. The similar setup for VMs in the cloud, e.g., virtualization techniques, VM OS, installed vulnerable software, networking, and so on, attracts attackers to compromise multiple *VMs*.

The evaluation is done by assigning to an individual a value that measures how suitable that individual is to the proposed problem. In our GP experimental environment, individuals are evaluated on how well they learn to predict good answers to a given problem, using the set of functions and terminals available. The resulting value is also called raw fitness and the evaluation functions are called fitness functions. Notice that after the evaluation step, each solution has a fitness value that measures how good or bad it is to the given problem. Thus, by using this value, it is possible to select which individuals should be in the next generation. Strategies for this selection may involve very simple or complex techniques, varying from just selecting the best n individuals to randomly selecting the individuals proportionally to their fitness.

3. METHODOLOGY

3.1 Cloud Components

A Cloud system consists of 3 major components such as clients, datacenter, and distributed servers. Each element has a definite purpose and plays a specific role.

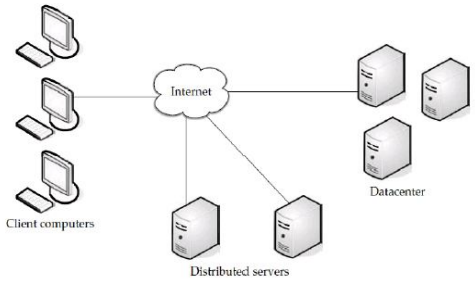

Figure 2: Three components make up a cloud computing solution(adopted from [1]).

Figure 3.1: Cloud components

Clients:

End users interact with the clients to manage information related to the cloud. Clients generally fall into three categories as given in:

- Mobile: Windows Mobile Smartphone, smartphones, like a Blackberry, or an iPhone.
- Thin: They don't do any computation work. They only display the information. Servers do all the works for them. Thin clients don't have any internal memory.

- Thick: These use different browsers like IE or Mozilla Firefox or Google Chrome to connect to the Internet cloud. Now-a-days thin clients are more popular as compared to other clients because of their low price, security, low consumption of power, less noise, easily replaceable and repairable etc. Datacenter.

Datacenter is nothing but a collection of servers hosting different applications. A end user connects to the datacenter to subscribe different applications. A datacenter may exist at a large distance from the clients.

Now-a-days a concept called virtualization is used to install a software that allow multiple instances of virtual server applications.

Distributed Servers:

Distributed servers are the parts of a cloud which are present throughout the Internet hosting different applications. But while using the application from the cloud, the user will feel that he is using this application from its own machine.

Services provided by Cloud computing:

Service means different types of applications provided by different servers across the cloud. It is generally given as "as a service". Services in a cloud are of 3 types as given:

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Hardware as a Service (HaaS) or Infrastructure as a Service (IaaS)

Software as a Service (SaaS)

In SaaS, the user uses different software applications from different servers through the Internet. The user uses the software as it is without any change and do not need to make lots of changes or doen't require integration to other systems. The provider does all the upgrades and patching while keeping the infrastructure running.

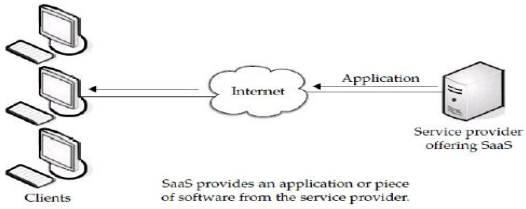

Figure 5: Software as a service (SaaS) (adopted from [1])

Figure 3.2: Software as a service

The client will have to pay for the time he uses the software. The software that does a simple task without any need to interact with other systems makes it an ideal candidate for Software as a Service. Customer who isn't inclined to perform software development but needs high-powered applications can also be benefitted from SaaS.

Customer resource management (CRM)

- Video conferencing
- IT service management
- Accounting
- Web analytics
- Web content management

Benefits: The biggest benefit of SaaS is costing less money than buying the whole application.

The service provider generally offers cheaper and more reliable applications as compared to the organization. Some other benefits include (given in): Familiarity with the Internet, Better marketing, smaller staff, reliability of the Internet, data Security[6], More bandwidth etc.

Obstacles:

- SaaS isn't of any help when the organization has a very specific computational need that doesn't match to the SaaS services
- While making the contract with a new vendor, there may be a problem. Because the old vendor may charge the moving fee. Thus it will increase the unnecessary costs.
- SaaS faces challenges from the availability of cheaper hardware's and open source applications.

Platform as a Service (PaaS):
PaaS provides all the resources that are required for building applications and services completely from the Internet, without downloading or installing a software.

PaaS services are software design, development, testing, deployment, and hosting. Other services can be team collaboration, database integration, web service integration, data security, storage and versioning etc.

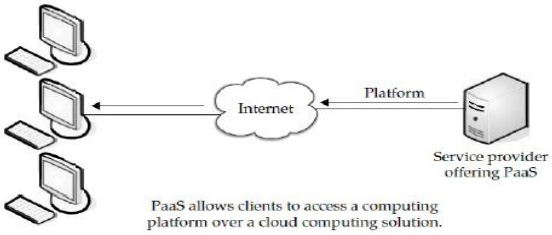

Figure 6: Platform as a service (PaaS) (adopted from [1])

Figure 3.3: Platform as a Service

Downfall:

- Lack of portability among different providers.

- if the service provider is out of business, the user's applications, data will be lost.

Hardware as a Service (HaaS):

It is also known as Infrastructure as a Service (IaaS). It offers the hardware as a service to a organization so that it can put anything into the hardware according to its will [1]. HaaS allows the user to "rent" resources (taken from [1]) as

- Server space
- Network equipment
- Memory
- CPU cycles
- Storage space

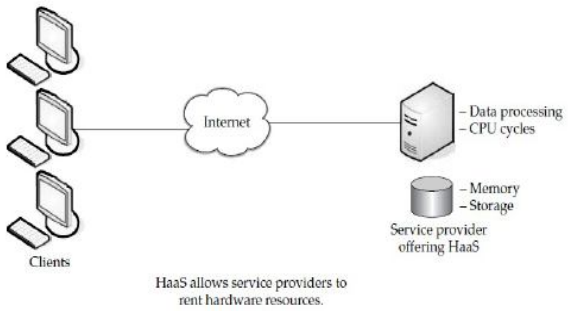

Figure 7: Hardware as a service (HaaS) (adopted from [1])

Figure 3.4: Hardware as a service

Cloud computing provides a Service Oriented Architecture (SOA) and Internet of Services (IoS) type applications, including fault tolerance, high scalability, availability, flexibility, reduced information technology overhead for the user, reduced cost of ownership, on demand services etc. Central to these issues lies the establishment of an effective load balancing algorithm.

4. IMPLEMENTATION

Design is concerned with identifying software components specifying relationships among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is minimal clearly specified.
During the system design activities, Developers bridge the gap between the requirements specification, produced during requirements elicitation and analysis, and the system that is delivered to the user.
Design is the place where the quality is fostered in development. Software design is a process through which requirements are translated into a representation of software.

4.1 Use Case Model

Use case diagrams represent the functionality of the system from a user point of view. A Use case describes a function provided by the system that yields a visible result for an actor. an actor describe any entity that interacts with the system. The identification of actors and use cases results in the definition of the boundary of the system, which is , in differentiating the tasks accomplished by the system and the tasks accomplished by its environment. The actors outside the boundary of the system, whereas the use cases are inside the boundary of the system

A Use case contains all the events that can occur between an actor and a set of scenarios that explains the interactions as sequence of happenings.

4.2 Java Programming Language

Each of the preceding buzzwords is explained in The Java Language Environment , a white paper written by James Gosling and Henry McGilton.

In the Java programming language, all source code is first written in plain text files ending with the .java extension. Those source files are then compiled into .class files by the javac compiler. A .class file does not contain code that is native to your processor; it instead contains bytecodes — the machine language of the Java Virtual Machine1 (Java VM). The java launcher tool then runs your application with an instance of the Java Virtual Machine.

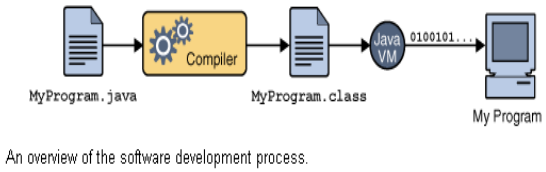

Figure 4.1: java software development process

An overview of the software development process.

Because the Java VM is available on many different operating systems, the same .class files are capable of running on Microsoft Windows, the Solaris Operating System (Solaris OS), Linux, or Mac OS. Some virtual machines, such as the Java HotSpot virtual machine, perform additional steps at runtime to give your application a performance boost. This include various tasks such as finding performance bottlenecks and recompiling (to native code) frequently used sections of code

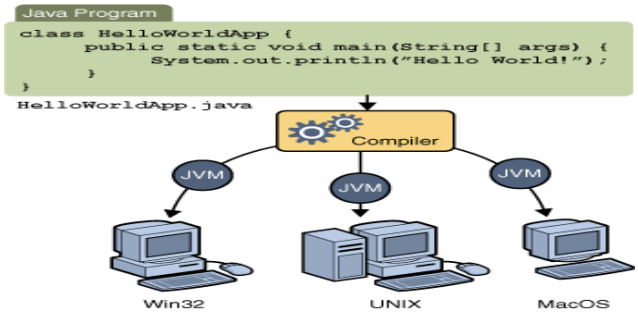

Figure 4.2: java compiler

Code Snippets (Logics) & Analysis

Logics

When using GP to solve a problem, there are some basic requirements that must be fulfilled, which are based on the data structure used to represent the solution. In our case, we have chosen a tree-based GP representation for the deduplication function, since it is a natural representation for this type of function. These requirements are the following:

1. All possible solutions to the problem must be represented by a tree, no matter its size.

2. The evolutionary operations applied over each individual tree must, at the end, result into a valid tree.

3. Each individual tree must be automatically evaluated.

For Requirement 1, it is necessary to take into consideration the kind of solution we intend to find. In the record reduplication problem, we look for a function that combines pieces of evidence.

In our approach, each piece of evidence (or simply "evidence") E is a pair <attribute; similarity function> that represents the use of a specific similarity function over the values of a specific attribute found in the data being analyzed. For example, if we want to reduplication a database table with four attributes (e.g., forename, surname, address, and postal code) using a specific similarity function.

To model such functions as a GP tree, each evidence is represented by a leaf in the tree. Each leaf (the similarity between two attributes) generates a normalized real number value (between 0.0 and 1.0). A leaf can also be a random number between 1.0 and 9.0, which is chosen at the moment that each tree is. Such leaves (random numbers) are used to allow the evolutionary process to find the most adequate weights for each evidence, when necessary. The internal nodes represent operations that are applied to the leaves. In our model, they are simple mathematical functions (e.g. *, % , /) that manipulate the leaf values.

To enforce Requirement 2, the trees are handled by sub tree atomic operations to avoid situations that could affect the integrity of the overall function, resulting an invalid tree. For a valid tree (or a valid function), there cannot be neither a case where the value of a leaf node is replaced by the value of an internal node nor one where the value of an internal node is replaced by the value of a leaf node.

According to Requirement 3, all trees generated during a GP evolutionary process is tested against pre evaluated data repositories where the replicas have been previously identified. This makes feasible to perform the whole process automatically, since it is possible to evaluate how the trees perform in the task of recognizing record pairs that are true replicas.

The tree input is a set of evidence instances, extracted from the data being handled, and its output is a real number value. This value is compared against a replica identification boundary value as follows: if it is above the boundary, the records are considered replicas, otherwise, the records are considered distinct entries. It is important to notice that this classification enables further analysis, especially regarding the transitive properties of the replicas.

4 This can improve the efficiency of clustering algorithms, since it provides not only an estimation of the similarity between the records being processed, but also a judgment of whether they are replicas or not.

After doing these comparisons for all candidate record pairs, the total number of correct and incorrect identified replicas is computed. This information is then used by the most important configuration component in our approach: the fitness function.

5. NICE OUTPUTS

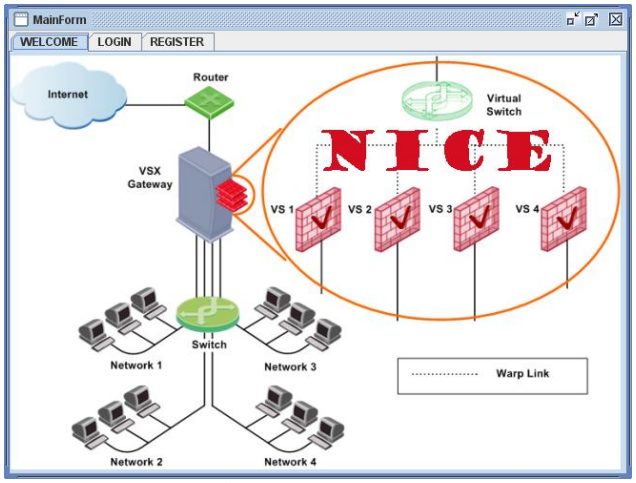

Figure 5.1:NICE

Login into NICE web-page

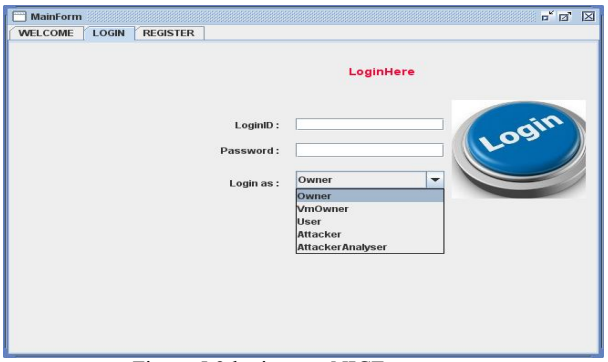

Figure 5.2:login page NICE

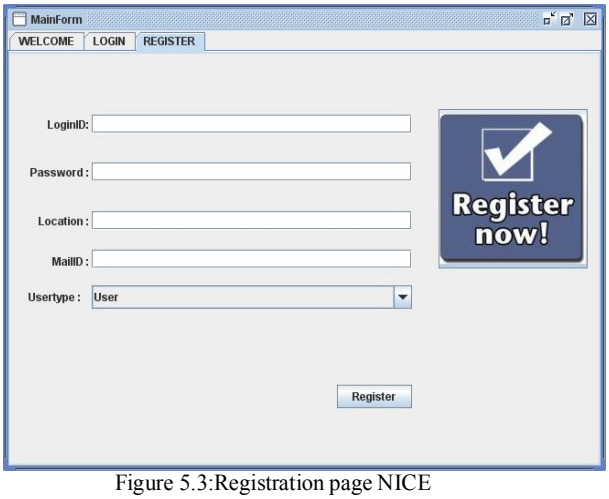

Figure 5.3:Registration page NICE

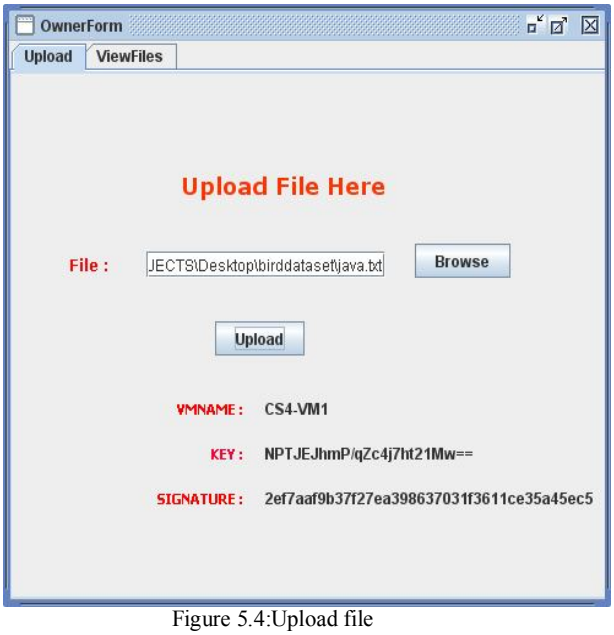

Figure 5.4:Upload file

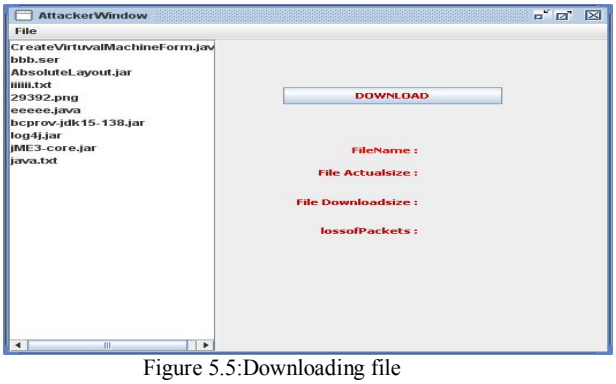

Figure 5.5:Downloading file

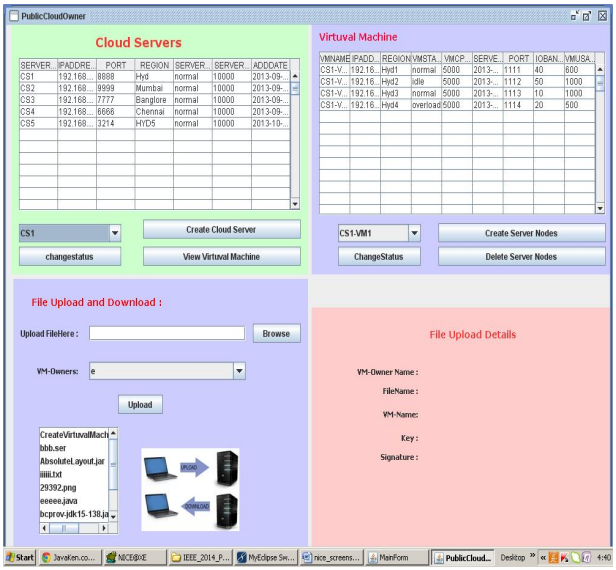

Figure 5.6: Intruder found the NICE

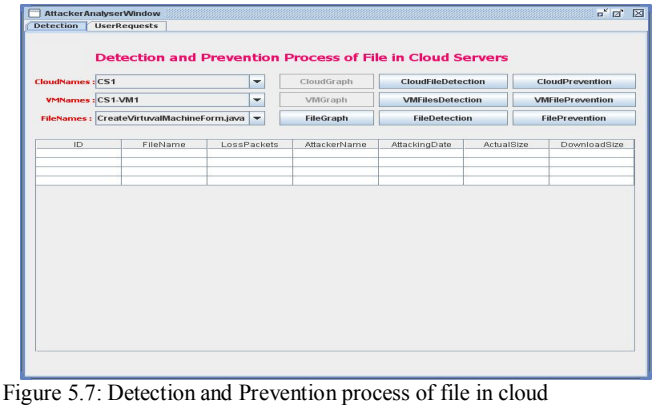

Figure 5.7: Detection and Prevention process of file in cloud services

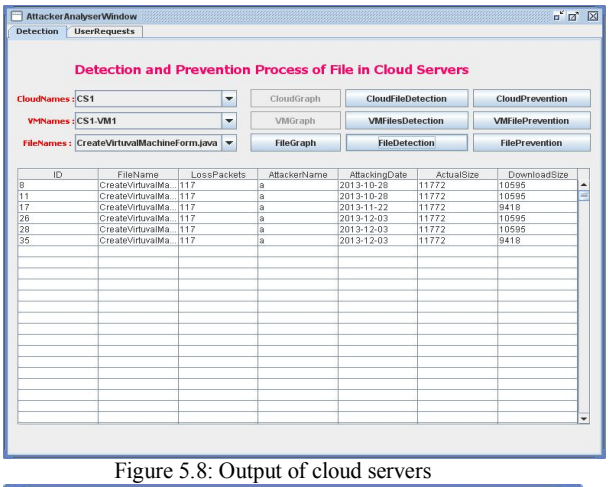

Figure 5.8: Output of cloud servers

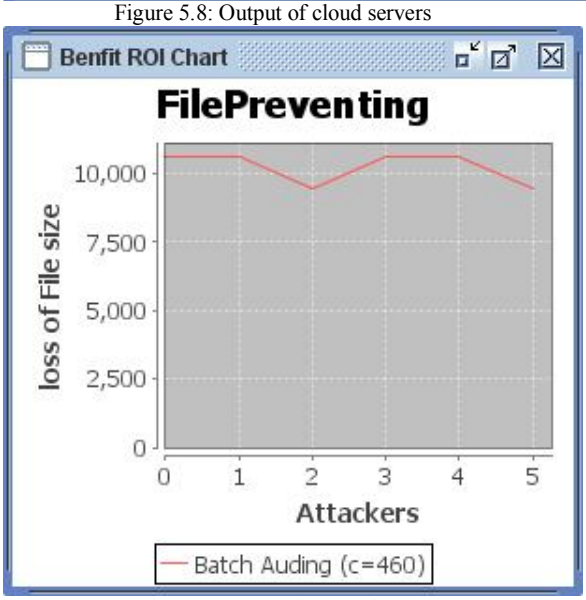

Figure 5.9: Benefit RIO chart

6. CONCLUSION

In this paper, we presented NICE, which is proposed to detect and mitigate collaborative attacks [9] 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 [7] 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.

7. FUTURE ENHANCEMENT

This should be investigated in the future work. Additionally, as indicated in the paper, we will investigate the scalability of the proposed NICE solution by investigating the decentralized network control and attack analysis model based on current study.

8. ACKNOWLWDGEMENTS

We are grateful to express sincere thanks to our faculties who gave support and special thanks to our department for providing facilities that were offered to us for carrying out this project.

REFERENCES

[1] Coud SercurityAlliance,"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 PrivacyChallenges 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] 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. 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. 15thAnn. 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. Security and Privacy, pp. 273-284, 2002,

[10] "NuSMV: A New Symbolic Model Checker," http://afrodite.itc. it:1024/nusmv. Aug. 2012.