# A Fuzzy Intrusion Detection System for Cloud Computing

Carolina Yoshico Ji[1], Nival Nunes de Almeida[2], Orlando Bernardo Filho[3]

[1]Globe Network Television, Rio de Janeiro, Brazil
[2]Rio de Janeiro State University (UERJ), Rio de Janeiro, Brazil
[3]Rio de Janeiro State University (UERJ), Rio de Janeiro, Brazil

***Abstract*—** *The aim of this work was to create an approach using fuzzy inference systems to detect intruders on cloud computing applications. Cloud computing is a topic that has been extensively discussed and although it is gaining market share, some researchers highlight a critical factor for their use, i.e., data security and reliability of processes. To validate the proposed approach, a distributed system was created in Java language in order to control a process of collaborative software development in the cloud. Two fuzzy inference systems were created: one to analyze network problems and another for analysis of security problems in distributed environment. Several tests were made in order to verify the operation and application of the proposed approach. Such tests were satisfactory.*

***Keywords*— Cloud Computing, Computer Network, Fuzzy Logic, Intrusion Detection System, Security.**

## I. INTRODUCTION

The increase of computer networks usage necessity leads directly to the complexity increasing of integrated management systems. This is due to increasing the number of new devices that are added to the network, and they need to communicate with each other, and thus the task of management [1] is becoming something every most important and crucial to their performance time.

Cloud computing has given help to this technological evolution process, but it also came with security problems to information exchange [2]. Initially, it may be understood as a form of information processing in which computing resources are highly scalable and they are offered to customers as services via Internet. Cloud computing is an all-inclusive solution where components such as hardware, software, networking and storage, among others, are provided to users quickly and on demand.

The network growth and complexity made the risks verification execution, in order to provide security to data and information, an essential task. Network traffic analysis to study and to investigate its behavior [3] is a way to ensure its integrity, avoiding attacks and suspicious events. In order to know if a network is protected or vulnerable to unauthorized access [4], the implementation of an Intrusion Detection System (IDS) is required.

Recently, on November 26th, 2012, some Google's services were disconnected during twenty minutes approximately; time enough to affect everyone [5]. The organizations using Google Apps, a tool to manage all e-mails of a company, were without e-mail service while failure was present. This episode showed the majority of countries have high international de-pendency on the services offered by Google. Thus, the fragility of the service became apparent. A question must be raised: to what extent can we fully trust on services using the cloud?

In this sense, the motivation for this work was to study the possibility of minimizing the network vulnerability and security risks in the cloud, by applying a proposed software development in the cloud. Data integrity is prime importance, and information systems are subject to physical and human failings and even constitute a potential target for malicious attacks for data acquisition.

## II. THEORETICAL BASIS

The demand for investment in information technology has be-come increasingly important and present at the organization's daily life. In this context, the computer networks expansion occurs because it is responsible not only to decrease the distance between the teams, but also the response time to re-quests. The computer networks popularity increase leads directly to complexity increase of management integrated systems. This occurs because more and more new devices entering the network, creating a need for communication between them, and thus, the network management task becomes increasingly essential and crucial to obtain good performance.

According to Saydam [6], the best definition of network management proposes that:

"*Network management involves the deployment, integration and coordination of all the hardware, software and human elements to monitor, test, poll, configure, analyze, evaluate, and control the net-work and element resources to meet the real-time, operational performance and quality-of-service (QoS) requirements at reasonable cost*".

In general, a security service is responsible for the events prevention that threaten the availability, scalability, integrity or confidentiality of a physical or computational resource by controlling, reducing or eliminating threats, risks and vulnerabilities. When a network security service is implemented, control mechanisms can be used aiming to prevent, to detect, or to recover from an attack. IDSs are classified as control mechanisms and attack detection. These systems have the functionality to detect attacks from malicious nodes and implement appropriate measures to maintain the operation of the network counter-measures. For technical reasons, the necessities of an Intrusion Detection System, at the computer networks context, are different for cloud computing.

At the present scenario, in conjunction with firewalls, IDSs have become a necessary component in most network security systems.

**Cloud Computing**

The use of the cloud computing term is becoming more common, with the promise of becoming a paradigm that will transform the current mode of development and marketing of software products. Referring to a recent term, there are still many doubts and disagreements about its meaning. Although there is no consensus on the concept of cloud computing, Foster *et al.* [7] provides an interesting definition and quite complete for this paradigm; it can be used as a reference for this job:

"*A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet.*"

The cloud is a metaphor for the Internet or communications infrastructure among architectural components based on an abstraction that hides the complexity of this infrastructure. Every infrastructure's part is provided as a service and this service is usually awarded in data centers using a common hardware to computing and storage [8].

The cloud computing solution offers lower costs, because users can access the services provided by cloud only with a computer configured with an operating system, a browser and network card (wired or wireless) to get Internet access. All computing resources are available in the cloud, and users do not have to worry about robustness computing services, resulting in the possibility of using inexpensive equipment. Cloud computing allows a high level of abstraction and customization, since any hardware can be used to realize a task suitable to its processing capacity. It is also possible add more advanced hardware and other devices in order to expand the capacity of processing and to cooperate with the existing resources.

For a long time, companies and data centers were built and designed so that the user did not need to have physical contact with computers, servers and disks, for example. This thought is founded on a security rule which states that if a person has the opportunity to have physical contact with a device; it can be more easily to damage such device. For many scholars and professionals of Information Technology (IT), the idea of putting applications in the cloud is alarming. When the cloud security issues are put in check, you need to think two or three types of threats. First, it is necessary to list the most common applications of cloud-based threats and the usual threats for those applications that are not in the cloud. The second list should focus in threats for specific applications running in the cloud.

## III. MODELING

In order to validate the approach cloud security with fuzzy logic, a case study that is designed to manage the development of a software product is implemented in a distributed manner. The environment considers several people (analysts and/or programmers) that interact via cloud to build a software product collaboratively. All members are coordinated by a manager who also supervises and evaluates the work of software development in the cloud.

The case study in question consisted of a distributed application to develop software. Each Node program (used by member) in the cloud would be responsible for creating the elements of a software design phase, for instance, during the analysis

phase, that could be performed by a member, the artifacts produced, e.g. diagrams and specifications, would be sent to the Manager program (used by software design manager). Figure 1 presents the case study architecture.

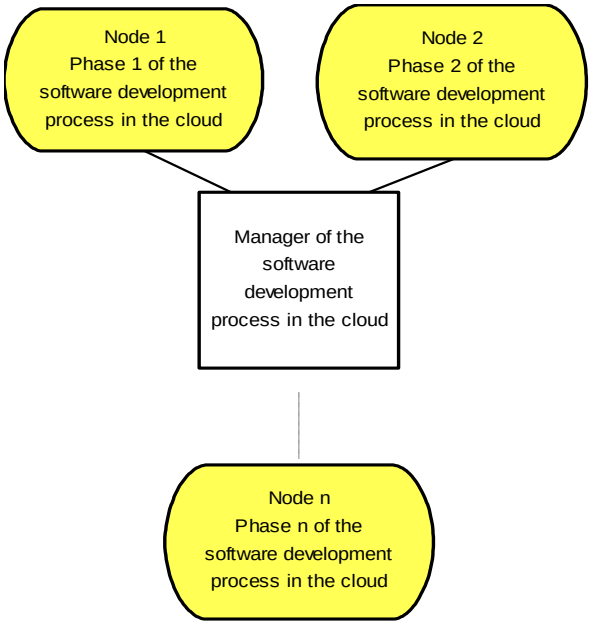


**FIG.1. DISTRIBUTED SOFTWARE DEVELOPMENT IN THE CLOUD SYSTEM ARCHITECTURE**

In this proposed approach, the cloud security is achieved by the use of two fuzzy inference systems that are executed by the Manager program. Such program manages the distributed system. This system is the case study. Both fuzzy inference systems are used to assess the security situation in the cloud, so that the distributed application will be considered safe when all nodes are safe. If a node has been attacked, it means that the application cooperative work in the cloud will not be completed.

The Software Manager in the Cloud (*softmancloud*) program is responsible for monitoring all nodes in the case study. It communicates with each Software Node in the Cloud (*softnodecloud*) program to infer if Nodes have or not a security problem. Since communication between Manager and Node can simply fail due to network problems without any security issue, it was designed two fuzzy inference systems: one to infer network problems, called Network Inference System (NIS), and another to deduce if there was really an attack on a Node, called Security Inference System (SIS).

The *softmancloud* program was developed according to object oriented programming good practice by using layers. It was used the Model, View, Controller (MVC) architecture, which helps maintenance and allows better structured project because there are fewer tanged code among program classes. Figure 2 shows the *softmancloud* class diagram where you can see the different layers shown in the program for each package:

**Packet user_interface** → contains classes' implementation responsible for the interaction with the user of *softmancloud* through windows and events perceived on objects displayed;

**Packet network_interface** → contains classes' implementation responsible for communication between *softmancloud* and the Software Node in the Cloud (*softnodecloud*) program;

**Packet controller** → contains classes' implementation responsible for the interaction among various layers, and it also has the task of promoting the full functionality of *softmancloud*;

**Packet model** → contains classes' implementation whose objects are responsible for keeping the data in memory for each record of each entity in the database used by *softmancloud*. These classes treat data from the database with a kind of auxiliary memory to receive them or to prepare them to be recorded in the database;

**Packet Data Access Object (DAO)** → contains classes' implementation responsible for implementing CRUD (Create, Retrieve, Update, and Delete) of tables' records in the *softmancloud* database.

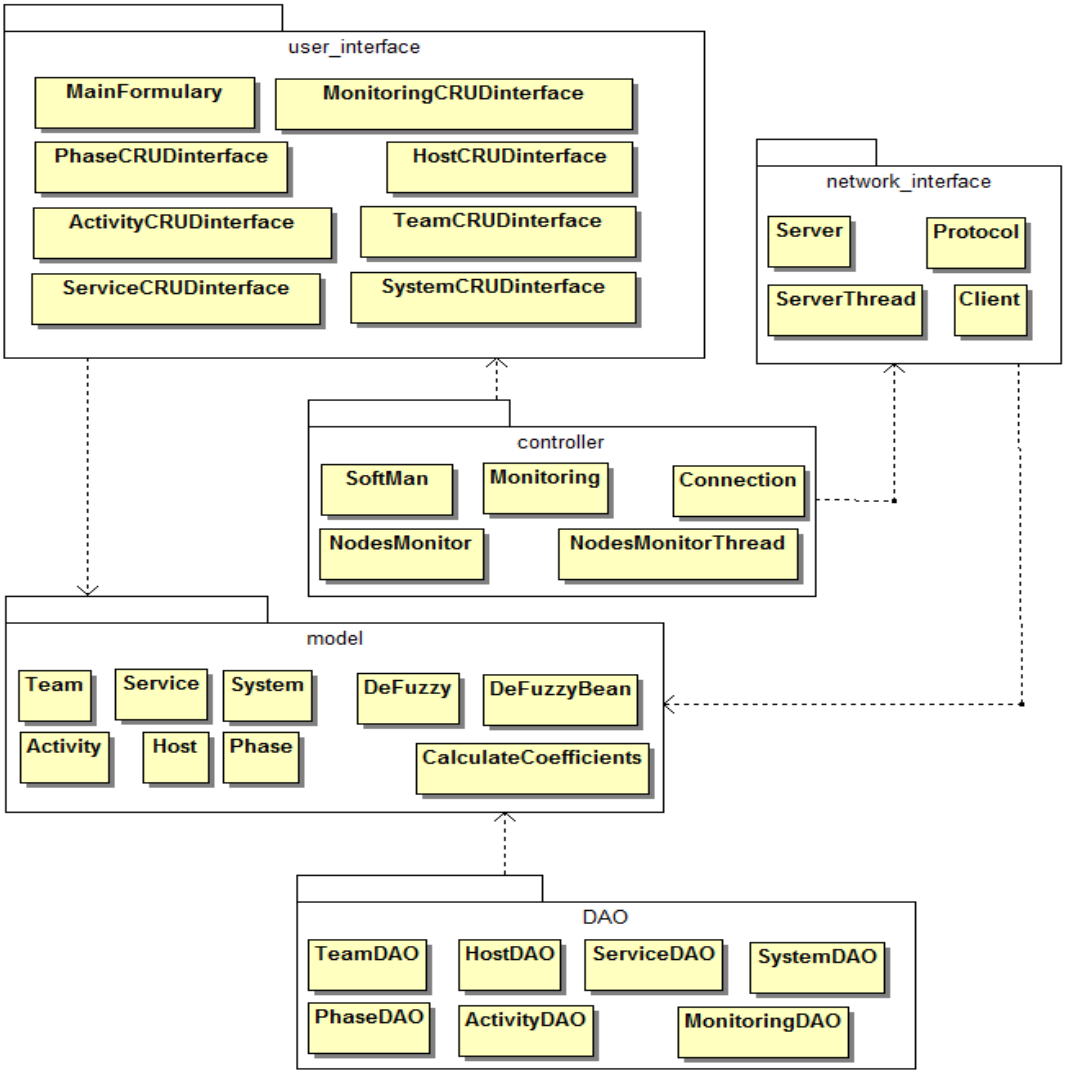


**FIG.2. THE SOFTMANCLOUD PROGRAM CLASS MODEL**

**Fuzzy Interference Systems Modeling**

The approach proposed for intrusion detection in this paper is based on the idea that an application running in the cloud must ensure that all nodes are running perfectly, so that collaborative work is effective. Therefore, this approach provides a monitoring of all components involved in the application in the cloud periodically, so you can detect a problem previously in any of these components, before critically compromising all the work.

Perform monitoring all nodes of the application in the cloud to ensure safety is exactly what characterizes the main contribution of the proposed model for intrusion detection, since most of the security mechanisms applied is implemented on a single host.

Whereas monitoring of cloud nodes involves network communication, it is necessary to rule out any network problems before concluding that a node has been attacked. In this case, the intrusion detection approach in the cloud comprises two fuzzy inference systems, shown in Figure 3:

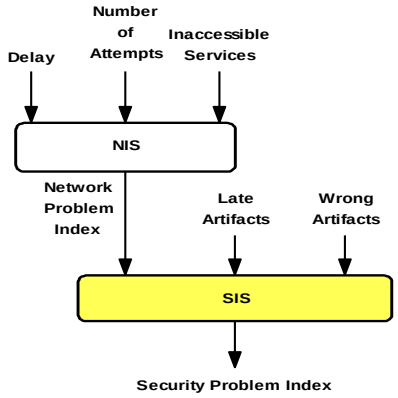


**FIG.3. FUZZY INFERENCE SYSTEMS FOR INTRUSION DETECTION IN THE CLOUD**

The reason for creation of these two fuzzy inference systems, specified by a security expert in computer networks, relies on the fact that some attack types, like deny of service (DoS), leave host without access to network. This means to isolate the host attacked and make it off-line. However, the host under analysis may not have access to network due to other problems (physical or software configuration), so it should be possible to differentiate how a host is inaccessible because it has a problem security or network problems.

The three input linguistic variables of NIS presented in Figure 3 are: Delay (D), Inaccessible Services (S), and Number of Attempts (A). The Delay is measured in milliseconds (ms), Number of Attempts is measured in number of times, and Inaccessible Services is measured in number of accessed services. Figure 3 also illustrates the output; the answer is the Network Problem Index (NPI) which indicates, in percentage, the possible existence of network problems on a node in the cloud.

The reason for using three input linguistic variables to SIS follow the concept expressed by expert. Initially, it was decided that there would be a separation between network problem analysis and security problem analysis, so naturally NPI would be one of SIS's input. After, it was considered by expert that some attacks, as Trojans, can tamper data from a host locally without affecting its operation, thus this would allow wrong data transmission. For this attack type, it is necessary to certify whether data received by manager are correct, thus it was created a linguistic variable named Wrong Artifacts (WA). Such variable will determinate how many received data by manager were wrong.

Finally, we must take into account the attack types as man in the middle, where exchanged data between two network elements, for example, manager and a node are intercepted by some way, stored and possibly tampered by intruder without victims' knowledge. The intruder, in this case, may decide subsequently retransmit data, but this data would generate a delay in the information delivery time. To determine such situation, it was created the SIS's input variable Late Artifacts (LA).

For example, they are presented the following NIS's inference rules:

IF **D** IS *short* AND **A** IS *little* AND **S** IS *nobody* **THEN** NPI IS *without problem*
IF **D** IS *medium* AND **A** IS *middle* AND **S** IS *somebody* THEN **NPI** IS *probable*
IF **D** IS *long* AND **A** IS *much* AND **S** IS *everybody* THEN **NPI** IS *with problem*

Similarly, we can highlight the following fuzzy inference rules of SIS, where its output is the Security Problem Index (SPI):

IF **NPI** IS *without problem* AND **LA** IS *little* AND **WA** IS *little* THEN **SPI** IS *without problem*
IF **NPI** IS *probable* AND **LA** IS *middle* AND **WA** IS *middle* THEN **SPI** IS *probable*
IF **NPI** IS *with problem* AND **LA** IS *much* AND **WA** IS *much* THEN **SPI** IS *with problem*

## IV. IMPLEMENTATION

The case study implemented uses two databases (softmancloud and fuzzylogic). They were created with the database management system MySQL [9]. The softmancloud database is responsible for maintaining the information necessary to control the development of a software product in the cloud, while fuzzylogic database contains data defining the fuzzy inference systems used by the case study.

The fuzzy inference algorithm follows the Larsen's model [10]. This model was chosen to allow the use of any member-ship function to the consequent of fuzzy inference rules, and it is also less complex to be implemented when compared to the similar Mamdani's model [11].

The class DeFuzzy.java contains the implementation of the fuzzy inference algorithm, and it uses, as defuzzification method, the arithmetic average of centroids weighted by area. A pseudo code for this algorithm is given below:

```
START INFERENCE (input1, input2, input3, system)
  sumCentroid = 0, sumArea = 0;
  ruleCentroid = 0, ruleArea = 0;
  SELECT Rules from System = system; // NIS or SIS
  FOR each Rule DO
    ruleCentroid = 0, ruleArea = 0;
    Get 1st Rule's linguistic value;
    p1 = Evaluate the membership function of input1 (1st value);
    Get 2nd Rule's linguistic value;
    P2 = Evaluate the membership function of input2 (2nd value);
    Get 3rd Rule's linguistic value;
    P3 = Evaluate the membership function of input3 (3rd value);
    Activation Degree = smallest value between p1, p2, and p3;
    IF Activation Degree > 0 THEN
      Get Consequent Value of Rule;
      ConsMod = Multiplier membership function of
                Consequent Value by Activation Degree;
      ruleCentroid = CalculateRuleCentroid (ConsMod);
      ruleArea = CalculateRuleArea (ConsMod);
      sumCentroid = sumCentroid + (ruleCentroid * ruleArea)
      sumArea = sumArea + ruleArea;
    END IF
  END FOR
  Final Value = sumCentroid / sumArea;
END INFERENCE
```

The inference algorithm presented above takes as arguments the three input values of system running (NIS or SIS), and so, it is calculated, for all fuzzy rules, the membership function value of each antecedent (present in rule) corresponding to respective input.

If smallest membership value is greater than zero, the algorithm proceeds to calculate: (1) centroid; (2) area, and (3) area x centroid product of membership function from consequent term multiplied by Activation Degree (smallest membership value among inputs). These calculated values are accumulated, and after processing all rules to each input received, the algorithm terminates with the calculation of the arithmetic average of centroids weighted by area. Such average is the inferred value by fuzzy system executed.

The monitoring algorithm implemented in Software Manager in the Cloud program is responsible for the periodic operation observation of nodes in the cloud. Such nodes process each phase of the software product that is in development. This algorithm makes the verification of nodes operation in respect to their functionality in the network and security, seeking to identify whether the queried Node is in perfect working condition.

The monitoring processes the analysis of each Node by running two fuzzy inference systems implemented (NIS and SIS). Each of these fuzzy inference systems is executed after calculate the values of its respective inputs.

During monitoring, the Manager program query various existing information from its databases, i.e., softmancloud and fuzzylogic. Figure 4 shows the activity diagram of the monitoring algorithm logic.

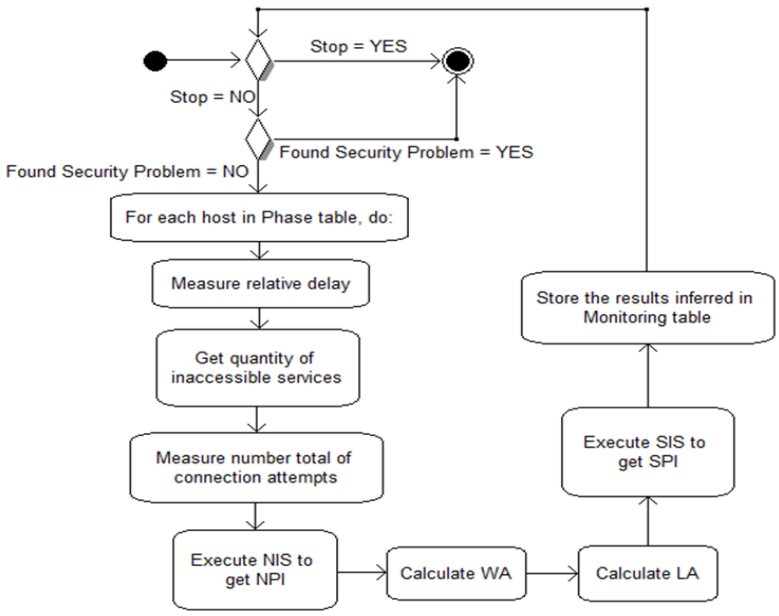


**FIG.4. THE *SOFTMANCLOUD*'S MONITORING ALGORITHM**

In Figure 4 is also possible to observe that the monitoring algorithm runs until the stop command is activated by the user or until a security problem in a Node is detected. NIS and SIS are executed during each algorithm's iteration for all nodes registered in database. NPI and SPI values inferred for each Node at each algorithm's iteration are stored in a softmancloud database table, called Monitoring, with Node's IP address and its inputs values Delay (D), Inaccessible Services (S), Number of Attempts (A), Wrong Artifacts (WA), and Late Artifacts (LA).

## V. TESTS AND RESULTS OBTAINED

For this work, it was prepared an environment containing four virtual machines. These machines represent the Nodes. The real machine represents Manager.

Tables 1 and 2 show the Manager's monitoring. In these tables, results of the four monitored Nodes are present (as indicated in column HOST), inputs measured values to run NIS and SIS are shown, and NPI and SPI values inferred are shown too. In this example, the monitoring was canceled after the second iteration to the Node with IP 192.168.56.103.

**TABLE 1. NIS EXECUTION DURING A MONITORING**

| HOST | D | S | A | NPI |
|---|---|---|---|---|
| 192.168.56.102 | 0.3135 | 0 | 1 | 16.6 |
| 192.168.56.103 | 10 | 0 | 1 | 49.9 |
| 192.168.56.104 | 10 | 1 | 10 | 50.0 |
| 192.168.56.105 | 10 | 2 | 10 | 50.0 |
| 192.168.56.102 | 0.0489 | 0 | 1 | 16.6 |
| 192.168.56.103 | 10 | 0 | 1 | 49.9 |

**TABLE 2. SIS EXECUTION DURING A MONITORING**

| HOST | NPI | WA | LA | SPI |
|---|---|---|---|---|
| 192.168.56.102 | 16.6 | 0 | 0 | 16.6 |
| 192.168.56.103 | 49.9 | 50 | 0 | 50.0 |
| 192.168.56.104 | 50.0 | 0 | 0 | 50.0 |
| 192.168.56.105 | 50.0 | 0 | 0 | 50.0 |
| 192.168.56.102 | 16.6 | 0 | 0 | 16.6 |
| 192.168.56.103 | 49.9 | 50 | 0 | 50.0 |

Table 3 shows the results of simulated attacks performed in tests conducted in this work. The types of attacks were:

**Spoofing:** This attack involves masking the IP address of a certain computer system. By hiding or faking a computer's IP address, it is difficult for other systems to determine where the computer is transmitting data from. This attack result in wrong artifacts created.

**Trojan:** Once installed, to map potential targets may hinder access to some services, which significantly increases the number of attempts and generate wrong artifacts.

**DoS:** Deny of Service has naturally cause delays in the packets that are in transit through the network, thus services remain inaccessible and produce late artifacts.

**TABLE 3. RESULTS OF SIMULATED ATTACKS**

| Attack | HOST | IPR | IPS |
|---|---|---|---|
| Spoofing | 192.168.56.102 | 16% | 50% |
| Trojans | 192.168.56.102 | 16% | 83% |
| DoS | 192.168.56.102 | 83% | 50% |

To simulate the spoofing, it was changed the IP address of a Node. Therefore, the NPI has detected that the target computer is not accessible, and the system indicates a network problem. To simulate Trojans, one machine is turned off and then it is assumed that the file came out of time and failed. To simulate the DoS, again one of the machines is turned off and the NPI has detected network problems.

## VI. CONCLUSIONS

The results obtained by Network Inference System (NIS) and by Security Inference System (SIS) generally have achieved the objectives, allowing to diagnostic if there were network problems or security problems.

Now, it is studying the possibility of some future work, being three quite promising: 1) create a system for verifying the integrity of transmitted artifact, avoiding the need of manual verification by a human; 2) develop a routine to receive problems notification via email, to be possible the manager be informed of the flaws in any place; and 3) expand the detection of security problems for other types of attacks, in addition to those that were tested.

The results obtained by Network Inference System (NIS) and by Security Inference System (SIS) served the purpose, and it was possible to detect the problems presented.

## REFERENCES

[1] M. P. Amaral and T. F. Faria, "Estudo comparativo entre as ferramentas CACTI e MRTG no gerenciamento de uma rede computacional com tráfego heterogêneo". Pós em Revista, ISSN 2176 7785, Edição 8, Novembro de 2013.

[2] H. Ruschel, M. S. Zanotto and W. C. da. Mota, "Computacão em Nuvem", 2010.

[3] J. M. A. Cardana, "Analisador Comportamental de Rede", Dissertacão (Mestrado) Universidade de Lisboa, 2006.

[4] T. H. Ptacek and T. N. Newsham, "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection", 1998.

[5] INFO, 2012. Available at http://info.abril.com.br/noticias/inter-net/usuarios-relatam-falhas-em-servicos-do-google-no-brasil-26112012-34.shl, 2012.

[6] Saydam and T. Magedanz, "Networks and Network Management into Service and Service Management", Journal of Networks and System Management, v. 4, n. 4, p. 345–348, December 1996.

[7] I. Foster, Y. Zhao, I. Raicu and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared". 2008.

[8] R. Buyya, R. Ranjan and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities", 2009.

[9] MYSQL, Available at http://www.mysql.com/, 2013.

[10] P. M. Larsen, "Industrial Applications of Fuzzy Logic Control", London: Academic Press, 1981.

[11] E. H. Mandani, "Application of Fuzzy Algorithms for Control of Simple Dynamic Plant", Proceedings of the IEE Control and Science, v. 121, p. 298-316, 1974.