International Journal of Advance Research in Science and Engineering
Volume No.07, Issue No.02, February 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

Security Intelligence in Cloud with Intrusion Detection and Prevention Enhancements

R.Sundar Raj[1], Dr.V.Murali Bhaskaran[2]

[1]Research Scholar, Research and Development Centre,
Bharathiar University, Coimbatore ,Tamilnadu,(India)
[2]Principal, Dhirajlal Gandhi College of Technology,
Omalur (Tk), Salem, Tamilnadu,(India)

ABSTRACT

Security is a two-sided coin in the world of Cloud Computing with its own pros and cons. It has some contentious issues in it, especially in the area of confidentiality and protection. The main situation and a common cause through which the cloud network becomes insecure, is the happening of intrusions. Intrusions that causes the downtime to the well-built systems are to be dealt with iron hands, both in case of paid services and free services, as well. To thwart the adversaries of the system and to mitigate the vulnerabilities, introducing an efficient Intrusion Controlling Systems both in terms of Detection and Prevention that associates well with the Cloud Resource Management is an essential factor. The previous works of the authors have proposed the systems String Based Intrusion Detection System (SBIDS) and Self Monitored Intrusion Prevention System (SMIPS). Also these systems' ideology and concept was conceived and the development methodologies was depicted with an analysis. In this research work, these systems with an innovative approach of seeking the support of string matching algorithms for the detection of intrusions and a self monitored approach in prevention of intrusions have been implemented. The presently followed Intrusion Detection and Prevention Systems in Cloud Computing may follow this proposed idea, to take out better results in identifying and preventing threat causing intrusions as early as possible with a lesser and easier effort in order to achieve the state of Security Intelligence.

Keywords: CSPs – Cloud Service Providers, EVMM- Enhanced Virtual Machine Monitor, Intrusion detection and prevention, SBIDS- String Based Intrusion Detection System, SMIPS – Self Monitored Intrusion Prevention System.

I.INTRODUCTION

The Cloud service generally starts from the Cloud Service Provider (CSP) and ends up at the user level. The cloud resource usually gets ordered by the user who may be an individual or an enterprise. In most of the cases, the resources are dispatched to the users with an utmost speed and with a least latency. This is carried out in order to keep up the customers in a satisfactory level, but this may lead to the problem of mishandling of resources with least security measures. Once the resource is given to the users, the usage starts and that is utilized by the real customers. Then after, the CSP would not interfere much in the usage of the resources, since it is a pay per use service and also due to the Service Level Agreements (SLAs) agreed. In some cases, these resources are used illegally by the intruders who may get in to the system due to mismanagement of resources at many levels like host, network or application.

Almost every main kind of attacks are caused due to the intruders. If the intrusion paths are abolished at the initial level of the service, then the intruders would not be able to initiate an attack on any level in the middle. The task of this kind, exists in the present level of cloud service, but not in an optimal manner and even if provided, they are costly to get affordable. In order to overcome these issues, the channels of cloud resource need not be demolished and reconstructed since they are in a good condition, but they need to be reconfigured in the necessitated areas by getting adaptation with some proven algorithmic approach that are available outside the cloud, in the general space. When these problems are addressed an end-to-end securely managed resource travels from CSPs to users, can be offered in low cost comparing to the present level.

1.1 Problems Addressed

From the wide spectra of Cloud, the following problematic areas have been proposed and implemented.

- Firstly, Security setup of the cloud needs focus. The chief attacks are caused due to the intruders. If the intrusion paths are not detected at Network level, then the attacks would reach the Host through Application. The main areas that are left unbothered are the conversion outcome of intrusion data packets, complex and lagging process of identification of new attacks. Thus, the need of a unique intrusion detection system for the cloud arises. Hence an improvised intrusion detection system has been implemented as String pattern Based Intrusion Detection System (SBIDS) in cloud
- Secondly, an exclusively compatible intrusion prevention system corresponding to the second portion of the work is highly needed, since these both are treated as twin systems. With an idea of addressing this need with proper extemporizations a biography based Intrusion Prevention System with rules through Security Intelligence in cloud in the third part of the work. This has been carried out by proposing Self Monitored Intrusion Prevention System (SMIPS).

II.RELATED WORK

Anup H. Gade in [1], has addressed the importance of cloud management solutions, by providing adequate resources for cloud applications. In [17], Rodrigo N. Calheiros et al., have taken efforts to design and develop Cloud technologies with Simulation-based approaches by using CloudSim toolkit. In [8], Dhinesh Babu L. D and Venkata Krishna have proposed a load balancing technique for cloud computing environments. Weiwei Kong et al., in [19], have proposed a work that is based on the traditional VM resource scheduling algorithms. Suraj Pandey et al., in [18], have presented and compared the results of a scheduling heuristic based on Particle Swarm Optimization (PSO), against Best Resource Selection (BRS) heuristic and claimed that PSO outperforms BRS. In [2], Asaju La’aro Bolaji et al., have proved that Artificial Bee Colony (ABC) has the ability of solving diverse sets of problems. Raj, R Sundar and Bhaskaran Dr. V Murali in [11], the technical guidance to overcome the major issues occurred on Cloud Computing has been proposed. Also, a roadmap to Virtualization approach in Cloud Computing has been proposed in [12]. In [13], an empirical study has been made with the motive of revolutionizing approach in Cloud Computing paradigm. Improved Cloud Security Mechanism with a Self-Monitored Intrusion Prevention System has been proposed in [14]. Dervis Karaboga and Bahriye Akay in their work [7], have compared the performance of ABC algorithm with those of Genetic Algorithm (GA), PSO, Differential Evolution (DE) and Evolution Strategy (ES) optimization algorithms and have proved that the ABC algorithm is better than others.

Raj, R Sundar and Bhaskaran Dr. V Murali, for the implementation of the intrusion controlling systems, the comparative analysis of cloud tools has been made in [15] to implement automated resource and security management system. Also in [16], securing cloud environment using a string based intrusion detection system has been proposed. In [10], Junaid Arshad et al., have focused on intrusion severity analysis in cloud and presented a Machine Learning (ML) based approach with Virtual Machine specific parameters. Chirag Modi et al., in [6] with their work have emphasized the usage of alternative options to incorporate Intrusion Detection System/ Intrusion Prevention System (IDS/IPS) into Cloud. In [3], Ashish Prosad Gope and Rabi Narayan Behera with their study, made it clear that Knuth-Morris-Pratt (KMP) algorithm performs better than other string pattern matching algorithms. In [5], B Rahul. Diwate and Satish J. Alaspurkar have projected that the performance of KMP has a less time complexity.

III.EXISTING INTRUSION CONTROL APPROACHES – A BRIEF ANALYSIS

The implementation is carried out at hardware level with cost demand and least flexibility in Physical Machines (PMs). In contrast to this, the implementation at software level with least cost and high flexibility in Virtual Machines (VMs). The common hurdles faced during the implementation of the existing IDS approaches are enlisted in Table 3.1.

Table 3.1 Difficulties observed in the Existing IDS Approaches based on Implementation

APPROACH	SETBACKS OBSERVED
Host based Detection	This is a single system based method and thus does not have a wide scope in the range of intrusion handling. This may have a very low scope.
	The chief problem in installing the agent of the intrusion control, the things that are to be matched with the factors such as the configuration of the system, the behaviour
	This method is very strict that it does not allow any changes made to the system, even in case of extreme necessity
Network based Detection	Since the network of cloud is scalable, the widening and shrinking of cloud range is constantly changing
	The compatibility of Network based and Signature based Intrusion Detection Systems goes tedious in some cases.

	The monitoring and updation of signatures is given by the service providers to the third parties of Network for the efficient monitoring scope, that arises questions on data security.
Application based Detection	The web applications are very easy to get implemented, which also facilitates the intrusions to get entered to the network.
	Setting-up the firewall for applications is somewhat tedious, since it should be implemented individually to the application level.
	The data generated as a result of intrusions by the applications are limited in their volume.

By comparing all these detection methodologies, some drawbacks are given below:

- The adaptation of Stack based anomaly management
- Performing the examination at the layers of Application and Protocol
- Capturing the whole traffic
- Configuring rule sets
- Insufficient handling of the Signalling mechanism
- Not considering all the categories of possible attempts

IV.PROPOSED INTRUSION CONTROL APPROACH

4.1 Knuth Morris Pratt (KMP) Algorithm

Knuth-Morris-Pratt (KMP) algorithm is an advantageous algorithm that is related to the task of matching the substring pattern to a main string pattern or the patterns that are available in the body of the text in a faster rate. This algorithm was first published in the year 1977 has got its name from the conceivers of the algorithm Donald Knuth, James H. Morris and Vaughan Pratt.

The highlighting part of the KMP algorithm is that it keeps track of the information of the nature of the pattern that are gained as a result obtained by the completion of previous comparisons. The 2n possible number of loops are iterated with three possible choices that are mentioned below in Table 4.1.

Table 4.1 Possible Choices on each iteration of KMP

CHOICE	OUTCOME WITH RESPECT TO CHANGES
T[i]==P[j]	The change of values: i++ and j, k remains unchanged
T[i]!=P[j], where j is +ve	The change of values: k++, j = f(j-1), and i remains unchanged
T[i]!=P[j], where j is 0	The change of values: i++, k++ and j remains unchanged *At beginning* : k = i – j *When j= f(j-1)* : k = i – f(j-1)

4.2 Prefix Table generation in KMP

The pseudocode of generating Prefix Table in KMP is presented in Fig. 4.1.

Prefix Table Generation

```
1.  begin
2.  m ← |p|
3.  Π[1] ← 0
4.  i ← 0
5.  for j= 2 until m do
6.      if i>0 and P[i+1] != P[j]
7.          i ← Π[i]
8.      else
9.          Π[j] ← i
10.   return Π
11. end
```

Fig. 4.1 Prefix Table Generation [KMP] - Pseudocode

4.3 String Pattern Matching

The pseudocode of matching the string pattern in KMP is presented in Fig. 4.2.

String Pattern Matching

```
1.  i ← 1
2.  j ← 0
3.  while i ≤ m-1
4.      if P[j] = T[j] then
5.          f(i) ← j + 1
6.          i ← i + 1
7.          j ← j + 1
8.      else if j>0 then
9.          j ← f(j – 1)
10.     else
11.         f(i) ← 0
12.         i ← i + 1
```

Fig. 4.2 String Pattern Matching [KMP] - Pseudocode

V. IMPLEMENTATION OF PROPOSED INTRUSION CONTROL APPROACH

5.1 Adaptability of KMP into SBIDS

The main objective of this part of the proposed approach is to overcome the setbacks of the surviving methods to carry out intrusion detection in improved means and by adapting KMP at the level of string pattern matching. This attempt takes solid steps to outstrip the existing works with its unique approach towards string conversion, matching, signalling, and reporting, by firming up the groundwork of the process. 'String Based Intrusion Detection System' (SBIDS) approach has been formed in a neat manner with its own flow of action. While formulating SBIDS much number of notations is used to carry out the process with the support of algorithm.

5.2 Objective Function and Key Constraints

The main objective is to maximize the detection possibility of the intrusions in a very earlier stage.

$$\text{Max} \sum_{k=1}^{n} D(I_k[t] + \mathrm{P}_k(\mathrm{ND}_k, \mathrm{LK}_k))$$

where, Trapped Packet index, k≥ 1; Ik[t] refers to the Intrusion suspect found at the time interval t; NDk refers to the Nodal points across the border zone of intrusion trap; LKk refers to the links associated to the nodal points.

Subject to the key constraints,

(i) Constraint of resource availability for carrying out the intrusion detection process

$$\sum_{k} RA_k\,(S_{Bik}, \mathrm{LK}_k) \geq \mathrm{I}_k\,[t] \qquad \forall\ \mathrm{SB}_{ik}, \mathrm{LK}_k, I_k$$

where, RAk refers to the resources availability of the system, SBik refers to the set of resources within the projected bandwidth of the detection area.

(ii) Speed of the detection while carrying out the regular tasks

$$\sum_{k} Sp\,(\mathrm{I}_k\,, S_{Bik}, \mathrm{LK}_k) \geq \mathrm{OA}(\mathrm{I}_k\,) \qquad \forall\ \mathrm{SB}_{ik}, \mathrm{LK}_k, I_k$$

where, Sp refers to the Speed of the detection over the intrusive attempt, OA(Ik) refers to the occurrence attempt of an intrusion with suspected intrusive properties

(iii) Not increasing the signalling rate for the unwanted interruptions due to the bad quality network packets

$$\sum_{k} SR\,(I_k, \mathrm{LK}_k) \leq {\sim}\mathrm{OA}(\mathrm{I}_k\,) \qquad \forall\ \mathrm{SB}_{ik}, \mathrm{LK}_k, I_k$$

where, ~OA(Ik) refers to the occurrence attempt of suspected intrusive properties due to the bad quality harmless packets

(iv) Not increasing the resource consumption further without letting down the capability of the detection mechanism.

$$\sum_{k} RC_k\,(S_{Bik}, \mathrm{LK}_k) \leq \mathrm{I}_k\,[t] \qquad \forall\ \mathrm{SB}_{ik}, \mathrm{LK}_k, I_k$$

where, RCk refers to the resources consumed for the detection process.

5.3 Solution Space

The solution space is equal to the range of the dynamically interconnected virtual space that is working with the physical network association with the demand to offer cloud resources. The scalability of the system has been verified in the distributed cloud environment. The screening of the packets is carried out in multi levels with the minimum of three (in terms of level – Network, Application, Host) and could be maximized based on the dynamic computation of inner sublevels of detection carried out. These two parts are intended in detecting and preventing intrusions with a string based approach, which is a pioneering idea and KMP which is a string matching algorithm is adapted to it. Appropriate biography information is maintained and duly updated regarding the intrusiveness and the actions that are to be taken are matched and followed with robustly framed rules. The intense of security is directly proportional to the value of an individual's or organization's resource. Hence the focus is channelized towards it.

5.4 Implementation - Cloud based Tools and Datasets

Appropriate cloud based tools for implementing the proposed approaches are speckled. For carrying out the resource related simulations and for dealing with the virtual elements of the system the framework of CloudSim is chosen and used. It has remarkable significances in fault tolerance, scalability and security with the support of modeling the application in the federated network of cloud. For carrying out the activities of intrusion detection and prevention the tool/ software named Snort is chosen and used in hand with CloudSim at the portion of creating rulesets.

Even though the operational data sets are available in many cloud locations as complementarily public, standard and key datasets from the repositories: University of California, Irvine (UCI) Repository and European Union Open Data Portal with academic disciplines [20][21] whose properties are listed in Table 5.1. They are exclusively chosen to inflow the datasets for the implementation of the proposed approach and to carry out the analysis.

Table 5.1 Properties of chosen Datasets

Data collection method	Document reviews and Surveys presented in Literature review
Data Source	UCI Repository, European Union Open Data Portal
Type of Attributes	Real
Data model, Parameter	Vector (1024)x10 parameters
Dataset characteristic	Multivariate, Time series
Instances, Statistical measure	Multiple Feature Selection, Standard Deviation
Normalized value of datasets	(Un-normalized value - Mean)/Standard Deviation
Critical value	In z-score by Normal Distribution calculator
MOE (Margin of Error)	Critical value x Standard Deviation
Nature of datasets	Machine Learning Community – Primary – Open Source
Missing data in the datasets	N/A
Sampling-Method;	Cluster Sampling; Normal

Distribution	
Standardization	Funded by National Science Foundation, Over 1000+ dataset citations in reputed journals

5.5 Snort – Rule based Implementation

Snort is a popular and an open source tool that is financially, technically and administratively easier to implement in small networks. The reason is that rules can be defined at the application layer of a packet in Snort which gives the possibility to collect traffic data, specifically in application layer. Also, Snort is useful to deploy the proposed SMIPS, with scalable stream classification framework with high accuracy and low runtime overhead, but may need some lag of processing time in classification that is not necessary, since the proposed approach is using matching, rather using classification. It is used to provide flexible, scalable, and a cost effective system for the cloud environment. The proposed system works on multiple logical layers of host or network or application level and as well as in platform and application levels.

5.6 Highlighting Factors of Implementation

It maximizes the security and detection accuracy, because it monitors every changes and traffic which is gone through each layer. The highlighting factor in this is that the new intrusion trying to enter into the network is stored in a separate entry log and get converted into strings and matched with the StrBuffer and if the match is not found in the string level, then it is tested at the substring level, if the match is found the whole system is protected with the necessitated action and the report is sent back to the network components and the entry is prohibited inside the network, but the whole converted string pattern is kept stored in the StrBuffer and action history is updated in ActBuffer. If a totally new kind of packet, if tries to enter the validation is done at multiple levels and demands the positive acknowledgment for the possible entry of the packet in to the system. Thus, SMIPS guarantees a high level security by entrusting principles in controlling and managing the intrusions, in a robust manner.

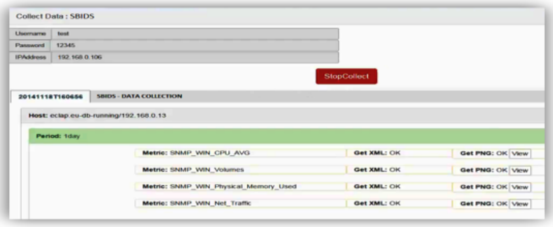
5.7 Implementation components

String matching would be the first consideration to dramatically improve the performance, as it accounts for about 75% CPU load of SMIPS. Thus, in this work multi-pattern string matching algorithm is proposed to examine such a traffic volume against a large set of strings. Moreover, certain signatures are represented in the regular expression to save the storage space, which may need pre-processing techniques to receive significant improvement. The implementations have given satisfactory performance.

VI.RESULT ANALYSIS

6.1 Implementation Results

(a)



Collect Data : SBIDS

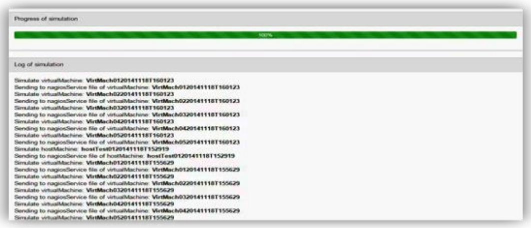
Username	test
Password	12345
IPAddress	192.168.0.106

Host: eclap.eu-db-running/192.168.0.13

Period: 1day

Metric: SNMP_WIN_CPU_AVG	Get XML: OK	Get PNG: OK View
Metric: SNMP_WIN_Volumes	Get XML: OK	Get PNG: OK View
Metric: SNMP_WIN_Physical_Memory_Used	Get XML: OK	Get PNG: OK View
Metric: SNMP_WIN_Net_Traffic	Get XML: OK	Get PNG: OK View

(b)



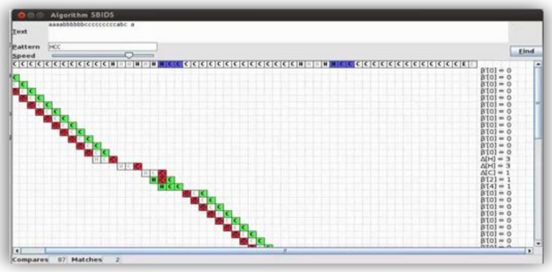
Progress of simulation

Log of simulation

```
Simulate virtualMachine: VirtMach0120141118T160123
Sending to nagiosService file of virtualMachine: VirtMach0120141118T160123
Simulate virtualMachine: VirtMach0220141118T160123
Sending to nagiosService file of virtualMachine: VirtMach0220141118T160123
Simulate virtualMachine: VirtMach0320141118T160123
Sending to nagiosService file of virtualMachine: VirtMach0320141118T160123
Simulate virtualMachine: VirtMach0420141118T160123
Sending to nagiosService file of virtualMachine: VirtMach0420141118T160123
Simulate virtualMachine: VirtMach0520141118T160123
Sending to nagiosService file of virtualMachine: VirtMach0520141118T160123
Simulate hostMachine: hostTest0120141118T152919
Sending to nagiosService file of hostMachine: hostTest0120141118T152919
Simulate virtualMachine: VirtMach0120141118T155629
Sending to nagiosService file of virtualMachine: VirtMach0120141118T155629
Simulate virtualMachine: VirtMach0220141118T155629
Sending to nagiosService file of virtualMachine: VirtMach0220141118T155629
Simulate virtualMachine: VirtMach0320141118T155629
Sending to nagiosService file of virtualMachine: VirtMach0320141118T155629
Simulate virtualMachine: VirtMach0420141118T155629
Sending to nagiosService file of virtualMachine: VirtMach0420141118T155629
Simulate virtualMachine: VirtMach0520141118T155629
```

Fig. 6.1 [SBIDS] (a) Collecting Data from Data Source, (b) Simulation Log – PMs and VMs

(a)



Algorithm SBIDS

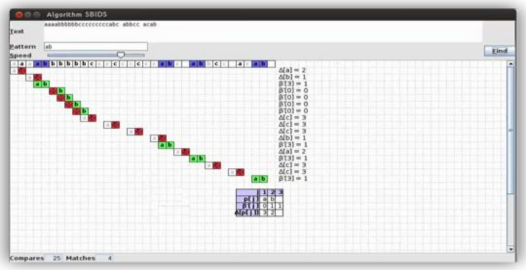
Text: aaaabbbbbbccccccccccabc a

Pattern: HCC

```
β[0] = 0
β[0] = 0
β[0] = 0
β[0] = 0
β[0] = 0
β[0] = 0
β[0] = 0
β[0] = 0
β[0] = 0
β[0] = 0
β[0] = 0
β[0] = 0
β[0] = 0
Δ[H] = 3
Δ[H] = 3
Δ[C] = 1
β[2] = 1
β[4] = 1
β[0] = 0
β[0] = 0
β[0] = 0
β[0] = 0
β[0] = 0
β[0] = 0
β[0] = 0
β[0] = 0
```

Compares 87 Matches 2

(b)



Algorithm SBIDS

Text: aaaabbbbbbccccccccccabc abbcc acab

Pattern: ab

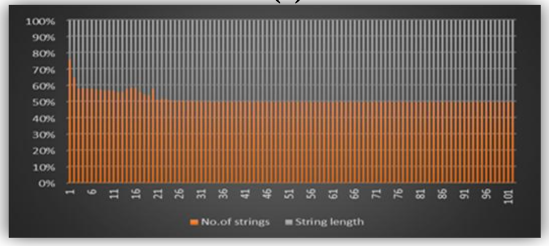
```
Δ[a] = 2
Δ[b] = 1
β[3] = 1
β[0] = 0
β[0] = 0
β[0] = 0
β[0] = 0
Δ[c] = 3
Δ[c] = 3
Δ[c] = 3
Δ[b] = 1
β[3] = 1
Δ[a] = 2
β[3] = 1
Δ[c] = 3
Δ[c] = 3
β[3] = 1
```

j	1	2	3
p[j]	a	b	
β[j]	0	1	1
Δ[p[j]]	3	2	

Compares 25 Matches 4

Fig. 6.2 [SBIDS] (a) Running the SBIDS Algorithm, (b) Match Log

(a)



(b)

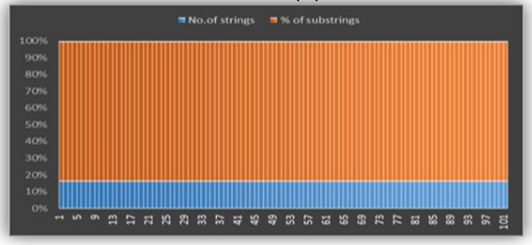
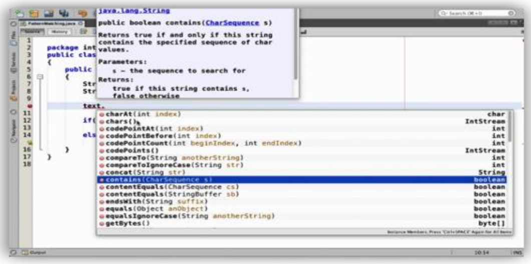


Fig. 6.3 [SBIDS] (a) String Match Rate, b) Sub string Match Rate

(a)

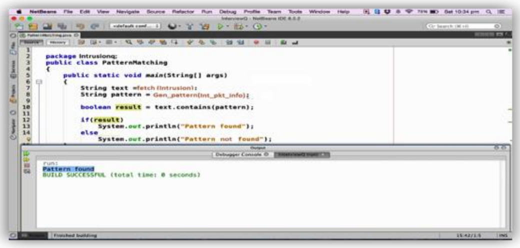


```
java.lang.String
public boolean contains(CharSequence s)
Returns true if and only if this string
contains the specified sequence of char
values.
Parameters:
    s - the sequence to search for
Returns:
    true if this string contains s,
    false otherwise
```

```
package int
public clas
{
    public
    {
        Str
        Str

        text.

        if(

        els

    }
}
```

```
charAt(int index)                                   char
chars()                                        IntStream
codePointAt(int index)                               int
codePointBefore(int index)                           int
codePointCount(int beginIndex, int endIndex)         int
codePoints()                                   IntStream
compareTo(String anotherString)                      int
compareToIgnoreCase(String str)                      int
concat(String str)                                String
contains(CharSequence s)                         boolean
contentEquals(CharSequence cs)                   boolean
contentEquals(StringBuffer sb)                   boolean
endsWith(String suffix)                          boolean
equals(Object anObject)                          boolean
equalsIgnoreCase(String anotherString)           boolean
getBytes()                                        byte[]
```

(b)

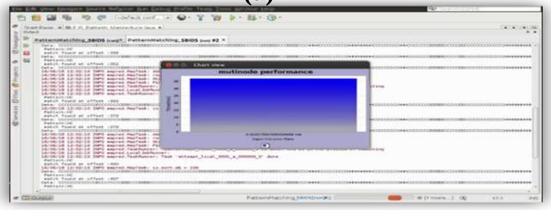

NetBeans File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Debugger Console
Finished building

```
package Intrusionq;
public class PatternMatching
{
    public static void main(String[] args)
    {
        String text =fetch(Intrusion);
        String pattern = Gen_pattern(Int_pkt_info);

        boolean result = text.contains(pattern);

        if(result)
            System.out.println("Pattern found");
        else
            System.out.println("Pattern not found");
```

```
run:
Pattern found
BUILD SUCCESSFUL (total time: 0 seconds)
```

Fig. 6.4 [SBIDS] (a) Matching the generated String Patterns, (b) Build Result

(a) (b)



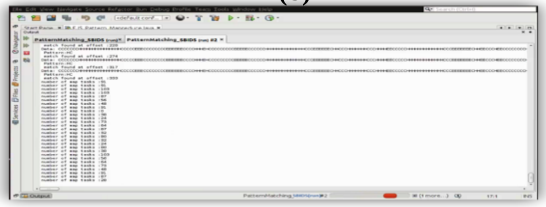
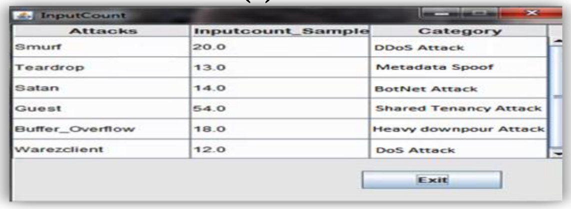


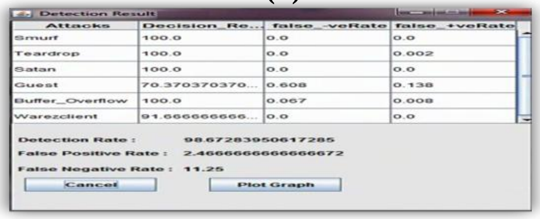
Fig. 6.5 [SBIDS] (a) Multinode performance in Pattern Matching, (b) Mapping Tasks

(a) (b)



InputCount

Attacks	Inputcount_Sample	Category
Smurf	20.0	DDoS Attack
Teardrop	13.0	Metadata Spoof
Satan	14.0	BotNet Attack
Guest	54.0	Shared Tenancy Attack
Buffer_Overflow	18.0	Heavy downpour Attack
Warezclient	12.0	DoS Attack



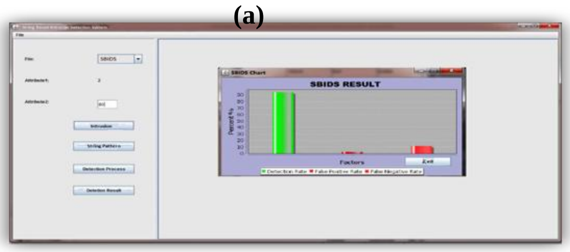
Detection Result

Attacks	Decision_Re...	false_-veRate	false_+veRate
Smurf	100.0	0.0	0.0
Teardrop	100.0	0.0	0.002
Satan	100.0	0.0	0.0
Guest	70.370370370...	0.608	0.138
Buffer_Overflow	100.0	0.067	0.008
Warezclient	91.666666666...	0.0	0.0

Detection Rate : 98.67283950617285
False Positive Rate : 2.4666666666666672
False Negative Rate : 11.25

Fig. 6.6 [SBIDS] (a) Input Count with category of Intrusions, (b) Detection Rate

(a) (b)



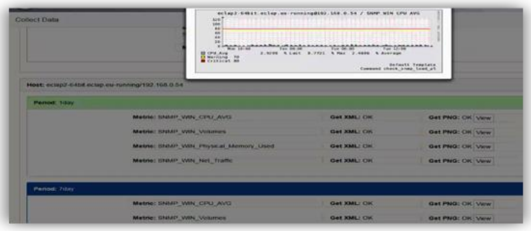


Fig. 6.7 [SBIDS] (a) Detection Result in Percentage, (b) Utilization of Processing Unit

(a) (b)

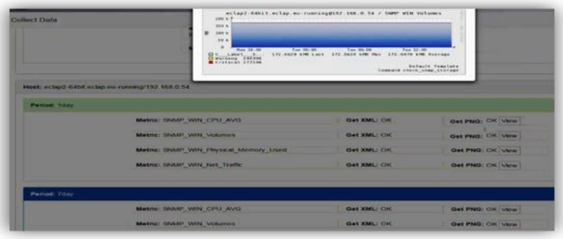
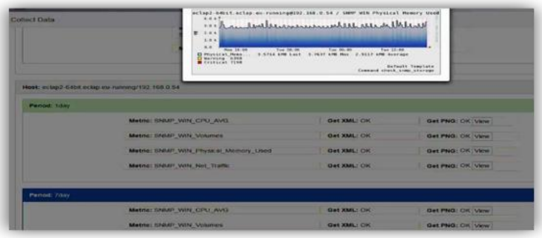



Fig. 6.8 [SBIDS] (a) Volume of Intrusion Data, (b) Utilization of Physical Memory

(a) (b)

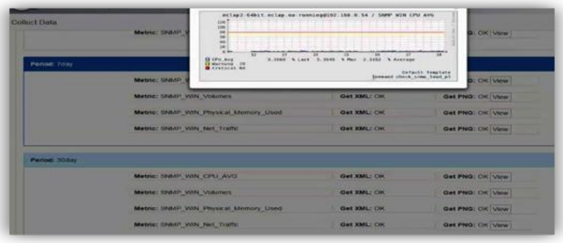
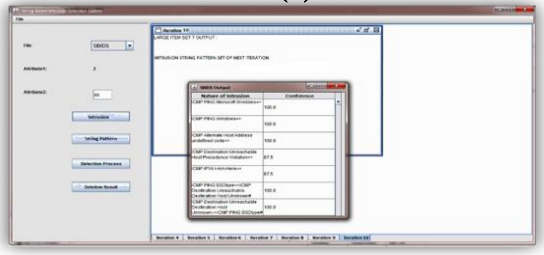
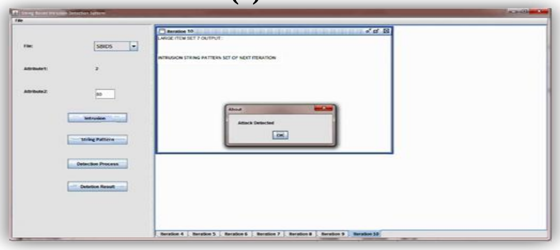
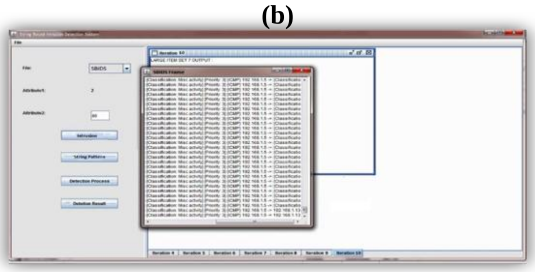



Fig. 6.9 [SBIDS] (a) Intensity of Network Traffic, (b) Intrusion String Pattern sets

(a) (b)



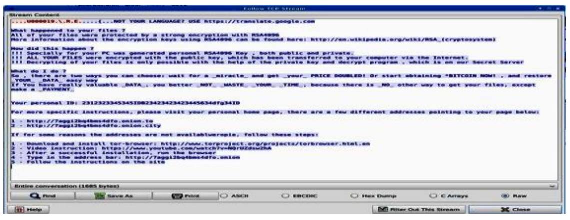
```
LARGE ITEM SET 7 OUTPUT :
INTRUSION STRING PATTERN SET OF NEXT ITERATION
Attack Detected
OK
```



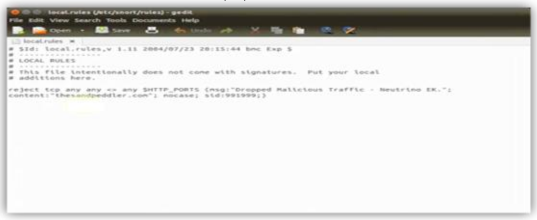
```
LARGE ITEM SET 7 OUTPUT :
```

Fig. 6.10 [SBIDS] (a) Detection of Attack, (b) Output of Itemset in Data Frame

(a) (b)



```
....U000019.\.R.E.....(...NOT YOUR LANGUAGE? USE https://translate.google.com

What happened to your files ?
All of your files were protected by a strong encryption with RSA4096
More information about the encryption keys using RSA4096 can be found here: http://en.wikipedia.org/wiki/RSA_(cryptosystem)

How did this happen ?
!!! Specially for your PC was generated personal RSA4096 Key , both public and private.
!!! ALL YOUR FILES were encrypted with the public key, which has been transferred to your computer via the Internet.
!!! Decrypting of your files is only possible with the help of the private key and decrypt program , which is on our Secret Server

What do I do ?
So , there are two ways you can choose: wait for a _miracle_ and get _your_ PRICE DOUBLED! Or start obtaining *BITCOIN NOW! , and restore
_YOUR_ _DATA_ easy way
If You have really valuable _DATA_, you better _NOT_ _WASTE_ _YOUR_ _TIME_, because there is _NO_ other way to get your files, except
make a _PAYMENT_

Your personal ID: 2312323345345I0B2342342342344563 4dfg34ID

For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below:

1 - http://7aggi2bq4bms4dfo.onion.to
2 - http://7aggi2bq4bms4dfo.onion.city

If for some reasons the addresses are not availablwerepie, follow these steps:

1 - Download and install tor-browser: http://www.torproject.org/projects/torbrowser.html.en
2 - Video instruction: https://www.youtube.com/watch?v=N0rUZdsw2hA
3 - After a successful installation, run the browser
4 - Type in the address bar: http://7aggi2bq4bms4dfo.onion
5 - Follow the instructions on the site
```


local.rules (/etc/snort/rules) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
local.rules

```
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.

reject tcp any any <> any $HTTP_PORTS (msg:"Dropped Malicious Traffic - Neutrino EK.";
content:"thesandpeddler.com"; nocase; sid:991999;)
```

Fig.6.11 [SMIPS] (a) Stream content of preventive action rules for users, (b) Defining Local rules

6.2 Analysis on Results

The SBIDS applications are facilitated in Fig. 6.1 (a), where the data collection happens with a basic login with Username, Password and additionally with the IP address. In Fig. 6.1 (b), the simulation log of PMs and VMs is shown through which the progress can be duly tracked. The numbers of compares and matches are shown in Fig. 6.2 (a), while running SBIDS with string pattern in speedy mode with the index values of arrays. At the same time, the match log associated with these compares and matches has been shown in Fig. 6.2 (b).

The match rate progress with number of strings against string length of main strings has been shown in Fig. 6.3 (a) and the match rate progress with number of strings against substring percentage of sub strings has been shown in Fig. 6.3 (b). In Fig. 6.4 (a), matching the generated string patterns has been shown with the code generation in NetBeans using java language with referred parameters and return values with the use of a built-in method contains (CharSequence s) that gives boolean results.

In Fig. 6.4 (b), the build log is shown along with the total time consumed to find the pattern match at debugger console, after the successful running the SBIDS algorithm. In Fig. 6.5 (a), the Multinode performance with time of detection against the memory consumed with the corresponding to the matching offset is shown with a chart view.

In Fig. 6.5 (b), mapping of tasks is shown with change of values in continuous mode. The input count with various categories of attacks such as Shared Tenancy attack, Downpour attack in heavy mode, Spoofing of Metadata attack, Botnet attack, DoS and DDoS attacks is shown in Fig. 6.6 (a) with input count samples respectively. In Fig. 6.6 (b), the False Negative and False Positive Rates have been successively given with the overall accumulated values of Detection Rate: 98.7 (approx. - Rounded), False Negative Rate: 11.3 (approx. - Rounded) and False Positive Rate: 2.5 (approx. - Rounded).

Among the files of the simulator, SBIDS application is opened as a new file, by setting necessitated attributes to carry out the detection process and a simple graph that has been generated to depict the obtained Detection Rate value with the values of False Negative and False Positive Rates is shown in Fig. 6.7 (a). Whereas in Fig. 6.7 (b), the overall utilization of processing unit is shown with command checks of sample loads and templates. The Volume of Intrusion Data that are trapped, the Utilization of Physical Memory and the Intensity of Network Traffic are shown in Fig. 6.8 (a), 6.8 (b) and 6.9 (a) respectively. In Fig. 6.9 (b), the iteration results are shown with the nature of the intrusions occurred along with the appropriate confidence values. The attack detection message popping out and the output of Itemset in Data Frame are shown in Fig. 6.10 (a) and 6.10 (b) respectively. Finally, in Fig. 6.11 (a) and 6.11 (b) the Stream content of preventive action rules for users, and results on defining local rules in Self Monitored Intrusion Prevention System (SMIPS) have been shown.

VII. CONCLUSION

By addressing these kind of intrusion based problems through this research work, an end-to-end securely managed resource travels from CSPs to users and can be offered in low cost comparing to the present level. The proposed approaches can give optimum solutions to the problems in the area of cloud resource and security management. It is to remark that any other cloud infrastructure similar to this, that emerges in future can be adapted and tested in the similar way. This is also open to improvisations for quenching any other future thirst in this area.

The experimental results show that the proposed dynamic resource allocation scheme can improve resource utilization and achieves a good performance with security over virtualized environments, that also reduces the energy consumption in a considerable amount with a social significance, thus reduces user's usage cost. Moreover, the proposed scheme has a simple implementation and, unlike many other approaches, it can avoid the complexity of re-allocation of physical resources with no intrusion possibility.

REFERENCES

1. Anup H. Gade. (2013). A Survey paper on Cloud Computing and its effective utilization with virtualization. *International Journal of Scientific and Engineering Research*, 4(12), 357-363.
2. Asaju La'aro Bolaji, Ahamad Tajudin Khader, Mohammed Azmi Al-Betar Mohammed A. Awadallah. (2013). Artificial Bee Colony Algorithm, its Variants and Applications: A Survey. *Journal of Theoretical and Applied Information Technology,* 47(2), 434 - 459.
3. Ashish Prosad Gope and Rabi Narayan Behera. (2014). A Novel Pattern Matching Algorithm in Genome Sequence Analysis. *International Journal of Computer Science and Information Technologies*, 5(4), 5450-5457.

4. Brototi Mondal, Kousik Dasgupta, Paramartha Dutta, Kalyani and Visva. (2012). Load balancing in Cloud Computing using Stochastic Hill Climbing-A Soft computing approach. *Elsevier, Science Direct- Procedia Technology, Bharati University,* 4(1), 783-789.
5. B Rahul. Diwate and Satish J. Alaspurkar. (2013). Study of Different Algorithms for Pattern Matching. *International Journal of Advanced Research in Computer Science and Software Engineering,* 3(3).
6. Chirag Modi, Dhiren Patel, Bhavesh Borisaniya, Hiren Patel, Avi Patel and Muttukrishnan Rajarajan. (2013). A survey of intrusion detection techniques in cloud, *Elsevier, Journal of Network and Computer Applications,* 36(1), 42-57.
7. Dervis Karaboga and Bahriye Akay. (2009). A comparative study of Artificial Bee Colony algorithm. *Elsevier, Journal of Applied Mathematics and Computation,* 214(1), 108-132.
8. Dhinesh Babu L. D and P. Venkata Krishna. (2013). Honey bee behavior inspired load balancing of tasks in cloud computing environments. *Elsevier, Applied Soft Computing,* 13(1), 2292-2303.
9. Fouad Bahrpeyma, Ali Zakerolhoseini, Hassan Haghighi and Shahid Beheshti. (2014). Using IDS fitted Q to develop a real-time adaptive controller for dynamic resource provisioning in Cloud's virtualized environment. *Elsevier, Applied Soft Computing,* 26(1), 285-298.
10. Junaid Arshad, Paul Townend and Jie Xu. (2013). A novel intrusion severity analysis approach for clouds. *Elsevier, Future Generation Computer Systems,* 29(1), 416-428.
11. Raj, R Sundar; Bhaskaran, Dr. V Murali. (2014). Technical Guidance to Overcome the Issues on Cloud Computing. *Journal of NanoScience and NanoTechnology,* 2(6), 725-727.
12. Raj, R Sundar; Bhaskaran, Dr. V Murali. (2015). A Roadmap to Virtualization Approach in Cloud Computing. *International Journal of Advanced Research in Data mining and Cloud Computing,* 3(1), 33-37.
13. Raj, R Sundar; Bhaskaran, Dr. V Murali. (2015). An Empirical study on revolutionizing approach in Cloud Computing paradigm. *International Journal of Contemporary Research in Computer Science and Technology,* 1(6), 187-198.
14. Raj, R Sundar; Bhaskaran, Dr. V Murali. (2016). Improved Cloud Security Mechanism with a Self-Monitored Intrusion Prevention System. *International Journal of Advance Research in Science and Engineering,* 5(12), 20-32.
15. Raj, R Sundar; Bhaskaran, Dr. V Murali. (2017). Comparative Analysis of Cloud Tools to Implement Automated Resource and Security Management. *Indian Journal of Engineering (An International Journal),* 14(35), 20-32.
16. Raj, R Sundar; Bhaskaran, Dr. V Murali. (2017). Securing cloud environment using a string based intrusion detection system. *Advanced Computing and Communication Systems (ICACCS), 20174th International Conference on*, 1-13.
17. Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, C´esar A. F. De Rose and Rajkumar Buyya. (2010). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *CLOUDS Laboratory,* 41(1),.23–50.

18. Suraj Pandey, LinlinWu, Siddeswara Mayura Guru and Rajkumar Buyya. (2008). A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments. *The University of Melbourne, Australia*.
19. Weiwei Kong, Yang Lei and Jing Ma. (2016). Virtual machine resource scheduling algorithm for cloud computing based on auction mechanism. *Elsevier, Journal of Optik*, 127(1), 5099–5104.
20. UCI Repository –
https://archive.ics.uci.edu/ml/support/Cloud
21. European Union Open Data Portal -
https://data.europa.eu/euodp/en/data/dataset/yUBHDpCh8MDqL9Gub8Qmq

AUTHORS' BIOGRAPHY

R.Sundar Raj, Research scholar in Bharathiar University, Coimbatore and Assistant Professor, Department of Computer Science in Kongu Arts and Science College (Autonomous), Erode, Tamilnadu, India has received his B.Sc in Computer Science and MCA degree from Bharathiar University and secured University I Rank in both the degrees. He has also published research papers in International journals. He is currently pursuing Ph.D programme with his area of research as Cloud Computing.

Dr. V. Murali Bhaskaran, Principal, Dhirajlal College of Technology, Omalur, Tamilnadu, India has nearly 30 years of experience in technical education. He obtained his B.E. Degree in Computer Science and Engineering from Bharathidasan University, Trichy in the year 1989, M.E. degree in Computer Science and Engineering and Ph.D in Computer Science and Engineering from Bharathiar University, Coimbatore in the year 2000 and 2008 respectively. He has published more than 40 papers in Journals and Conferences both at National and also in International level. His areas of interest include Computer Architecture, Computer Networks, Network Security, Information Security, etc.,