DOI 10.29042/2018-3188-3197

Helix Vol. 8(2): 3188-3197

Anomaly Detection System in Cloud Computing Network

***[1]S.Bharath Reddy, [2] D.Malathi, [3] S.Shijoe Jose**

[1, 2, 3] SRM Institute of Science and Technology, Tamil Nadu, India

***Email:** bharath .bittu945@gmail.com*

Received: 20th December 2017, Accepted: 20th January 2018, Published: 28th February 2018

Abstract

Security in cloud computing poses higher challenges due to modern day distributed computing architectures. Cloud computing is a quickly developing IT show for the trade and conveyance of various administrations through the Internet. Be that as it may, there are a plenty of security worries in cloud computing which still should be handled (e.g., confidentiality, auditability and Privileged User Access). To recognize and avoid such issues, the Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) are powerful instruments against assaults, for example, SQL Injection. Monitoring such a highly complex network environment needs a system capable of handling passive monitoring efficiently with less human intervene. Intrusion detection is a versatile security paradigm which can avert most of the computer and network related attacks if efficiently employed. In this paper, we propose a modern security monitoring framework capable of detecting security attacks through data intelligence. The results of experiments prove that the average time taken to predict attacks by K-means algorithm is better than Naive Bayes, MLP, and SVM algorithms.

Keywords: Intrusion Detection, Intrusion Prevention, Network Anomaly detection, K-means, Packet splitting, Duplicate Insertion, Payload mutation.

Introduction

Cloud computing has become ubiquitous computing platform for on-demand and cost-effective infrastructures. The exceptional effectiveness of the platform arises from the technology to share the underlying physical resources not limited to CPU Cores, RAM, Storage and Networking between users at the same time without compromising the performance. With the advent of emerging virtual machine technologies, cloud computing has a stronger growth in broad adoption. Virtualization technologies play a key role in fuelling the rapid growth of cloud computing platform. Virtualization allows resources on the single physical machine to be abstracted by a virtual layer enabling to group multiple virtual resources on different physical machine hardware's as one massive virtual resource. Physical devices in a cloud computing infrastructure are connected with high speed wired links. When a new virtual machine is created with some specific resource requirement, the allocated virtual resource could be from different physical machine based on the virtual machine placement algorithms[1].

Deployment strategies and different virtualization technologies possess various limitations and also open new areas of challenges for infrastructure reliability. Operation of cloud computing infrastructure mainly relays on networking of multiple physical resources. Due to the distributed computing resources, it is evident that a vulnerable virtual machine could significantly affect the overall cloud computing infrastructure performance. Network security for cloud computing environment possesses a significant degree of the challenge due to distributed nature of the infrastructure. In this paper, we explore the vulnerabilities that degrade the performance of cloud computing platform concerning the underlying networking infrastructure and proposed a framework for network security deployment using K-Means clustering algorithm.

Type of attacks

Insider attack

Supported Cloud clients possibly will push to get (and use wrongly) unapproved benefits. Insiders may perform cheats and reveal estimations to others (or change data deliberately). This addresses a veritable sureness issue. For instance, an inside DoS strike aligned with the Amazon Elastic Compute Cloud (EC2) [1]

Flooding attack

In this strike, the assailants try to surge difficulty by conveyance a tremendous number of gatherings from a guiltless host (zombie) in the system. Packets can be of sort TCP, ICMP, UDP or a blend of them. This sort of trap might be achievable because of unlawful system affiliations

For a circumstance of Cloud, the requesting for VMs are open by anyone through the Internet, which may achieve DoS or DDoS [2] assault by means of zombies. The flooding assault influences the administration's acquired by approved client. By assaulting a solitary server giving an individual administration, an assailant can bring about lost open door for the proposed benefit. Such an assault is called coordinate DoS assault [3]. On the off chance that the server's equipment resources are totally depleted by handling the surge asks for, the diverse administration cases on the related equipment machine are no more extended ready to finish their expected undertakings. Such kind of assault is called aberrant DoS assault. The flooding assault may raise the utilization charges to a great degree as the Cloud would not have the capacity to separate between the ordinary use and fake use.

Port scanning

Port checking [4] gives a quick overview of open ports,

close ports, and sifted ports. Through port filtering, aggressors can get open ports and strike associations running on these ports. Compose related particulars, for example, IP address, MAC address, switch, entryway secluding, firewall principles, and so on can be seen through this assault. Unmistakable port taking a gander at systems are TCP checking, UDP isolating, SYN filtering, FIN isolating, ACK checking, Window investigating and so forth [5]. In Cloud situation, an aggressor can strike offered benefits through port checking (by finding open ports whereupon these associations are given).

Backdoor channel attacks

It is an inactive assault which gifts programmer to increase secluded access to the tainted hub to trade off client secrecy. Utilizing secondary passage channels [6], the programmer can control casualty's assets and can make it as a zombie to offer DDoS assault. It can likewise be connected to uncover the classified information of casualty. Because of this, the bargained framework confronts trouble in working its normal undertakings.

Firewall (in Cloud) could be the ordinary clarification to keep a portion of the assaults noted previously. To stay away from ambushes on VM/Hypervisor, inconsistency based interference area systems can be related. For flooding strike and discretionary passage channel assault, moreover signature based impedance conspicuous confirmation or anomaly based interruption territory technique can be second-hand.

Although numerous IDPS frameworks have been proposed, their proper arrangement and control for successful assaults detection/prevention action and effective assets utilization have dependably been testing [7], [8]. The assessment of the IDPS execution for any given security design is a critical advance for enhancing their continuous capacity [9]. Another worry is identified with the effect of security implementation levels on the execution and ease of use of an undertaking data framework. In this paper, we consider the effect of security requirement levels on the execution and ease of use of an endeavor data framework. Specifically, we dissect the effect of designing an IDPS run checking process alongside its resulting activity (i.e. caution or drop) on the subsequent security of the system, and on the normal administration time per occasion.

Evasion Techniques in Cloud Computing

Five common techniques can evade the examination of an IPS: (1) denial-of-service (DoS), (2) packet splitting, (3) duplicate insertion, (4) payload mutation, and (5) shellcode mutation. We will introduce each of them as follows.

Denial-of-service: the first will be the refusal denial-of-service (DoS) attack, which anticipates that will overwhelm organize information transmission or schema assets, for example, the cpu and the memory space of the IPS [10]. Other than making a generous volume for framework traffic, those strike could ill-use those inadequacy of the distinguishment calculations. An instance done [11] exhibits that it will be possible to significantly again off those oversee facilitating figuring over grunt 2. 4. 3 by regulating those illumination sort out traffic on ill-use the mossy cup oak negative situation execution of the wrist bindings facilitating over Snort, which uses backtracking attempting with spread know possible sample coordinates over a run those hint at. Main 4. 0kbps of exchange velocity to those regulated information traffic may be sufficient to the DOS attack. This algorithmic multifaceted nature ambush may be a sample for dos assaults without eating up helter skelter framework exchange pace.

Packet splitting: It joins IP break and TCP/IP division, hacks IP datagrams alternately TCP/IP stream under non-covering segments or sections, especially negligible ones. If an IPS doesn't reassemble those IP bits alternately TCP/IP segments should re-set up those central requisition content, it might overlook a strike installed under the substance focused on the setback need. For example, an IPS might search for a check/canister/sh in the pack payload. Make that Likewise it may, those assailant might deliberately part those payload for the register with two areas, particular case containing/container and the different containing/sh. In the event that those IPS doesn't reassemble those two areas, it will a chance to be momentous to find those stamp over possibly bit with the objective that those forcefulness could stay away from its affirmation. Since the IPS screens every last one of traffic through the skeleton under its supervision, ideally, it necessities should reassemble every last one of IP parts What's more TCP/IP areas in the traffic on counter possibility avoiding. Regardless, an IPS might need obliged structure favourable circumstances for screen per-affiliation data [12] (e.g., those TCP/IP states and the reassembled requisition content) for a gigantic skeleton. For example, those space flowed should a help for reassembly might be insufficient. In this way, a strike may, regardless, need an opportunity to show up in the IP territories or TCP/IP isolates that an IPS happens not to reassemble.

Duplicate insertion: Duplicate insertion is a technique done which aggressors implant duplicate or coating segments (or IP sections) with perplex the IPS. This system depends upon that those IPS and the casualty might manage the copy/covering ends alternately segments12 conflictingly on account of those IPS needs related data, for example, sort out topology and the casualty's working skeleton.

Coating IP pieces and TCP/IP segments camwood be ambiguous will an IPS. For instance, expect that a TCP/IP part conveys those arrangement amount 10 and the substance ATTXYZ, What's more another area to a comparative cooperation conveys the grouping number 13 and the substance ACK. Those setback group will

interpret those provision substance Likewise whichever attack or ATTXYZ following getting those two sections, unforeseen upon the victim's working schema [13]. Without comprehending the working framework, an IPS will most likely interpret those provision substance conflictingly for the setback. An aggressor along these lines can use the equivocalness to dodge the identification.

For an immaculate world, chief's camwood configure those plan will set how the useful skeleton around each host in the inward framework will interpret the parcels for a particular uncertainty, for those objective that those IPS camwood interpret those approaching packs dependably. Manual configuration may be screw up inclined, with the goal those partake) energizes [14] proposes a changing mapping technique to test every host Furthermore determine the approach proficiently. A couple components might convolute the mapping done act. To instance, accomplice an IP convey with a host will be not facilitated with those use for NAT what's more DHCP. The changing trying might a chance to be questionable due to pack filtering Eventually Tom's perusing firewalls, alternately astounding package drops with respect to An partly switch when those traffic volume through it will be helter skelter.

Payload mutation: Payload mutations intimates an assailant progressions pernicious bundle payloads under semantically tantamount ones. The transformed payloads will show up should be interesting starting with the denote that an IPS expects, so that those strike could dodge the distinguishment. Since the semantics of the transformed payloads may be those same, the attack will be so far great of the setback. For instance, an ambush on the uniform asset Identifier (URI) for an http request might make transformed under a couple changed articulations using those libwhisker library (www. wiretrip. Net/rfp/txt/whiskerids.html). The change might a chance to be URI hexadecimal corrosive encoding, self-reference catalogs, turn around traversal indexes et cetera (see the A while ago said Web page for those focuses from claiming interest) to talk of the URIs for a couple semantically indistinguishable twin structures. Take URI hexadecimal corrosive encoding for example. On the off opportunity that those list name CGI-receptacle popular may be encoded Likewise %63%67%69%2d%62%69%6e, identikit the mark CGI-canister will misfire In the changed portrayal will be not institutionalized under CGI-container in the recent past the IPS distinguishment.

A string from claiming SQL implantation might similarly be encoded also to shirking. Similar to standardization to duplicate inclusion, Institutionalization for payload transform could a chance to be ambiguous clinched alongside light of the truth that the objective hosts might get ready those substance previously, an unforeseen way, unforeseen upon those provisions. For instance, an apache Web server doesn't recognize angled punctuation lines concerning illustration bona fide cuts, yet rather An Microsoft IIS server does. In this manner, executive's requirement should configure the IPS should recognize over those provisions on the objective need and bring a predictable viewpoint of the requisition payloads as those objectives.

Shellcode mutation: Shellcode mutation encodes a shellcode (i.e., a bit of code to abuse product helplessness) into polymorphic structures to avoid an IPS that identifies a shellcode as indicated by the marks extricated from one or a couple of variations of that shellcode. A few strategies need aid possible for those polymorphism. For instance, a forcefulness camwood encode or pack the shellcode, Also prepend a bit of code with unscramble or decompress the shellcode in the enterprise. An attacker could similarly supplant a bit of the 1st code with various, notwithstanding semantically proportionate rules. An inconsequential body of evidence in the final one the event may be embeddings those nop directions, i.e., no operation, will settle on those code show up to a chance to be interesting. A guideline, say mov eax, ebx, might be also supplanted for two directions push ebx and pop eax, for example. Since those Stamp to the shellcode doesn't appear in the polymorphic frame, those IPS will disregard on recognizing it. The frameworks are similarly discovered done pernambuco wood projects, for example, infections what more worms is. The writers clinched alongside [15] advertised an Audit about such shirking methods. Since there is an over the top amount of methodologies to change a shellcode, recognizing polymorphic codes on an IPS will be particularly doubtful. An IPS might require to unscramble those mixed code should re-establish those principal signature, or considerably duplicate those code execution (e.g., for a sandbox that imitates those execution on the goal hosts) with find pernambuco wood behavior. The undertakings from claiming re-establish those shellcode semantics on-line need aid in this way computationally exorbitant also issue those stack from claiming an IPS.

Challenges in Network Security on Cloud Computing

Conventional network security devices including but not limited to Firewalls, IDS are entrusted for infrastructure security.

Cloud computing infrastructure possess a new challenge on the deployment model of security devices within cloud premises. Cloud computing infrastructure provides scalable on-demand computing resources, reducing infrastructure costs for users. Dynamically scalable infrastructure is facilitated with advancements in virtualization and physical hardware management platform modules. Bare metal cloud service provides complete physical machine as a resource to user. In bare metal cloud services one physical NIC is utilized to serve multiple users with separate set of network policies such as IP address, bandwidth and firewall rules. Virtualized cloud service provides virtual resources to users by abstracting physical machine

resources through virtualization tools such as VMware, Openstack & Linux KVM. Single NIC will be shared by multiple users whose virtual machines reside on the same physical machine sharing resources such as CPU, RAM and Storage. Cloud deployment model which is shown in Figure 1 could have distributed resources separating the infrastructure as a distributed platform for storage, computing and networking. Such deployment models create more complicated network topology, which become a challenge for deploying security devices within the cloud infrastructure. In traditional computing infrastructure model network security devices such as Firewalls, IDS/IPS are middle boxes residing near the boundary of connectivity to the internet. Conventional network security device placement worked well to a certain degree in cases where a single user owned a complete physical machine. But in a cloud computing model, the same user may share the physical machine with many number of users and also may have multiple services on multiple virtualized physical machines evading the boundary existed in early computing infrastructure models. This new model lead to complex network topology where it become hard to employ security policies. Advanced cloud infrastructure technologies even provide rich set of features like migrating virtual machine from one location to another location in real time with minimal service disruption but without impacting the service to users. These Cloud characteristics such as live migration, portability, region based cloud servers, isolation and virtual machine customization has created more chaotic network design & management (2). With more chaotic network topology, traditional perimeter based defence solutions which acts as a boundary between internal and external networks tend to fail. To counter such scenario, various approaches have been pursued by researches, out of which cluster based IDS/IPS deployment become widely adopted.

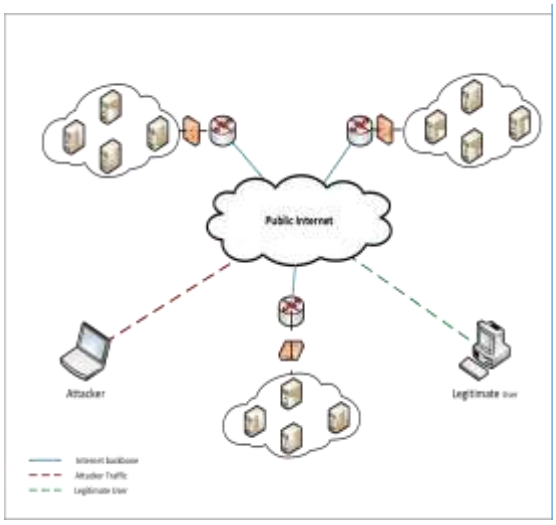


Figure 1: Generic Cloud Deployment Model

In a cluster based IDS/IPS deployment scenario, each and every machine which resides on the cluster may perform one or more operations collaboratively. But the key drawback on this method is that individual flow has to be processed across different machines on the clusters creating bottleneck for delay sensitive traffic.

Machine Learning based IDS using K-Means

Network security is very crucial for cloud infrastructure. A cloud user providing service to customer may have more Inter data centre data transfer because multiple services running on various machines to provide one service to user. At the same time there could be a malicious server which disrupts network performance creating more delay or blackout communication.

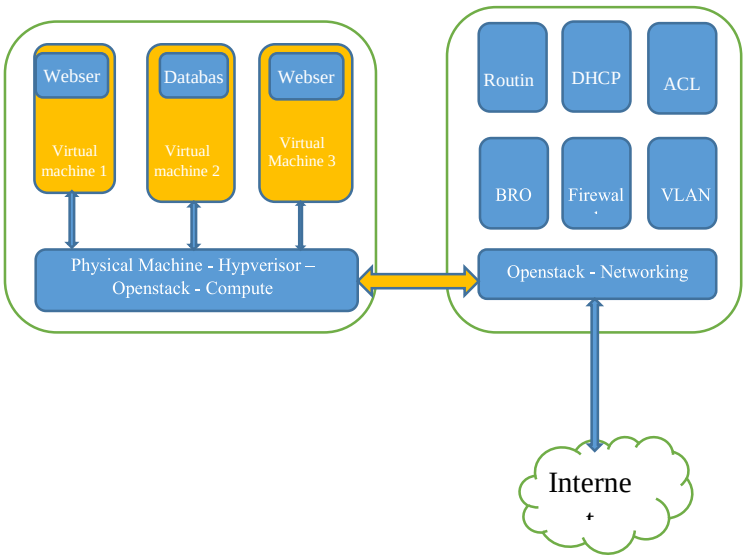
So, it is essential to have a security enforcement mechanism that will protect cloud infrastructure from premises attacks and attacks which alleviate from internet. General network security mechanisms such as perimeter firewall control, access control list, distributed firewalls & IDS/IPS are playing major roles in infrastructure security. The aforementioned mechanisms provide infrastructure security based on predefined or known attack vectors. Predefined attack vectors could be comprised of virus or malicious traffic signatures that defines the degree of attacks. Devices employing such techniques will look for the match of a signature with the packet content on the network. If the signature matches then an action will be taken either to block the IP or Port number to which the attack was targeted. Although these devices provide higher level of security for known attack vectors, they tend to fail when the attack vector is changed. Increase in sophisticated network attack tools lead to more diversified attacks which are hard to define their vectors. In order to identify and detect such attacks, it is necessary to create a system that will learn the attack based on the observation of the underlying network traffic patterns. In other words, we need to create a system that over time could justify the network behaviour as normal or anomalous with the help of observed network statistics.

A Network Anomaly Detection System (NADS) is a multistage system which includes several phases of data collection, pre-processing, data transformation, feature selection and clustering. In data collection phase, NADS will passively sample the data passing on the network. Sampled data will be pre-processed to identify the useful information for the rest of the operation without effecting the outcome of the experiment. Transformation is the process in which the sampled data could be scaled or normalized while preserving the data features. Transformed data may have some features which are not of highly important. In feature selection process, those features which are not useful will be removed leaving only a subset of the available features or sampled data. In the clustering phase of NADS system, sampled data will be grouped and labelled. Though all the above mentioned process must be the key phase of clustering in the performance of NADS.

In this work, K-Means clustering algorithm was implemented with Euclidean distance as a similarity

measure for evaluation of the network traffic developed as a module on Bro Network Security Framework which is shown in Figure 2. The objective of K-Means algorithm is to partition the traffic set in to a fixed number of disjoint subsets and then locate the cluster centre. Network connections will be clustered based on 5-tuple information found on the packet header by measuring the similarity between each connection. Once a cluster centre is located, then it will reestimate the similarity of connections and this time there may be a change in cluster members. For every phases of similarity measure, the cluster will locate its new centroid.

Figure 2: Testbed Architecture Bro Network Security Framework



Let the set of flows on the network be represented as

$F = f_1; f_2; f_3:::::f_n$

and k - number of clusters to be created on the set of flows

f_i - Sequence of IP packets exchanged between a pair of end points

The distance measure should meet the following criteria:

1. The distance between f_1 and f_2 should always be greater than or equal to zero.

$$dist(f_1; f_2) >= 0 \quad (1)$$

2. The distance between f_1 and f_2 is equal to zero if and only if f_1 is equal to f_2.

$$dist(f_1; f_2) = 0 \text{ if } f_1 = f_2 \quad (2)$$

3. The distance between f_1 and f_2 is equal to the distance between f_2 and f_1

$$dist(f_1; f_2) = dist(f_2; f_1) \quad (3)$$

The objective function of the system is as follows:

$$NADS(O) = \sum_{K=1}^{c} \sum_{i=1}^{n_k} \| f_i^k - C_k \|^2 \quad (4)$$

- Step 1: *Initial cluster centroid is fixed randomly over the set of flows for C clusters*
- Step 2: *Calculate the mean for each cluster C_i*
- Step 3: *Assign each flow f_i to the nearest cluster C_i centroid based on Euclidean distance between current cluster location and mean of the different clusters*
- Step 4:*Move f_i to nearest cluster*
- Step 5: *Reestimate and the new cluster centre and reassign the flows f_i*
- Step 6:*Repeat the steps from 1 to 5 until cluster won't change*

Experimental Setup

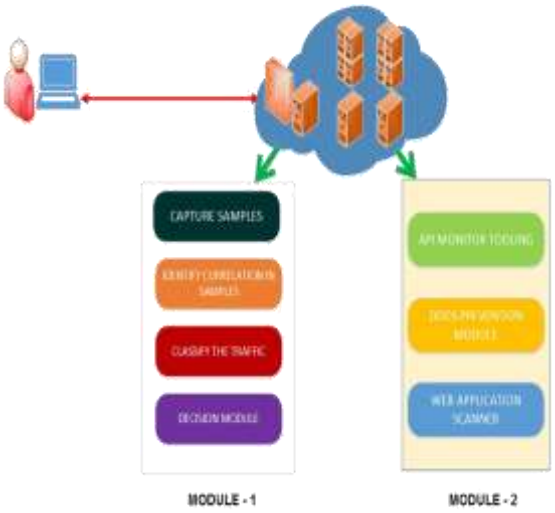
The experimental setup which is shown in Figure 3 consists of Openstack (1) cloud with 3 compute nodes and 1 controller node. In this system compute node also acts as a storage and network client controlled by the openstack Network control node.

Bro security monitoring framework is deployed as the traffic monitoring tool. Bro is an opensource network security framework which keeps track and logs all the events on the network. It is also a modular framework with advanced features such as traffic inspection, attack detection with capability for distributed analysis.

A security framework on observing network events, will log network activity based on the logic programmed on the logger module. The logger module provides rich set of API's to define the network activity that needs to be logged. We developed custom module for traffic classification which classifies the packet as malicious or genuine based on the logs created by logger module. Based on the outcome of the traffic classifier module firewall rules can be programmed to allow or deny a particular flow.

The proposed solution is validated with the latest dataset for DDoS attacks. In this work,

Figure 3: Bro Security Framework Architecture of Module Implemented



the Dataset from Canadian Institute of Cybersecurity (CIC) was used for clustering. The traffic traces on the dataset is exactly for 24 hours. The dataset also provided an XML file with flow information for each day. Dataset contained XML file containing the following field values:

"appName"; "totalSourceBytes"; "totalDestinationBytes"; "totalDestinationPackets";"totalSourcePackets"; "sourcePayloadAsBase64"; "destinationPayloadAsBase64";"destinationPayloadAsUTF "; "direction"; "sourceTCPFlagsDescription";"destinationTCPFlagsDescription"; "source"; "protocolName"; "sourcePort"; "destination"; "destinationPort"; "startDateTime"; "stopDateTime"; "Tag"

The flows have been generated using IBM QRadar appliance. The"Tag" column indicates whether the flow is normal or part of an attack scenario.

Table 1: Data sets collected from Canadian Institute of Cyber security

Friday	11/6/2010	Normal Activity. No malicious activity	16.1 GB
Saturday	12/6/2010	Normal Activity. Contains a small number of bruteforce attacks	4.22 GB
Sunday	13/6/2010	Infiltrating the network from inside + Normal Activity	3.95 GB
Monday	14/6/2010	HTTP Denial of Service + Normal Activity	6.85 GB
Tuesday	15/6/2010	Distributed Denial of Service using an IRC Botnet + Normal Activity	23.4 GB
Wednesday	16/6/2010	Normal Activity. Contains a small number of bruteforce attacks	17.6 GB
Thursday	17/6/2010	Brute Force SSH + Normal Activity	12.3 GB

The dataset contained information we couldn't use it out of the box. The dataset had a lot of features but the only one which is useful for the purpose were 5-tuple information consisting of: source ip address - ip address of the computer from which the packet is generated

source port - port number of the computer from which the packet is generated destination ip address ip address of the computer to which the packet was delivered destination port number - port number of the com puter to which the packet was delivered Time - time at which the packet was received protocol - type of packet tcp , udp or icmp

Table 2: Sample Data's with attributes

SOUCE IP ADDRESS	SOURCE PORT NUMBER	DESTINATION IP ADDRESS	DESTINATION PORT NUMBER	DATE RECEIVED	PROTOCAL	TYPE
115.178.60.136	18	10.35.22.161	56	11/6/2010	TCP	NORMAL
10.35.22.181	116	115.146.70.121	85	11/6/2010	UDP	NORMAL
71.45.142.70	96	15.23.146.173	74	11/6/2010	TCP	ATTACK
10.35.22.181	256	115.178.60.136	68	12/6/2010	UDP	NORMAL
116.144.60.136	74	176.83.122.162	129	12/6/2010	ICMP	ATTACK
94.135.64.187	714	154.12.58.136	523	13/6/2010	UDP	ATTACK
115.146.25.168	564	84.16.45.152	945	13/6/2010	ICMP	NORMAL

122.35.14.36	12	154.32.158.16	65	14/6/2010	UDP	ATTACK
25.13.45.126	78	78.12.45.123	38	14/6/2010	TCP	NORMAL
145.23.65.84	45	65.23.54.168	745	15/6/2010	UDP	ATTACK
184.16.23.177	67	115.45.23.89	648	15/6/2010	UDP	ATTACK
158.56.15.156	89	186.97.114.23	69	16/6/2010	TCP	NORMAL
86.14.28.119	345	172.145.23.56	38	16/6/2010	UDP	NORMAL
145.36.58.79	206	13.58.45.123	483	17/6/2010	ICMP	ATTACK
89.145.46.17	69	153.1.46.18	629	17/6/2010	TCP	NORMAL

After processing the data we calculated the entropy for those data using bit shifting method.

The code for bit shifting is as follows:

```
public long ipToLong(String ipAddress)
{
long result = 0;
String[]ipAddressInArray=ipAddress.split("\\.";
for (int i = 3; i >= 0; i--)
{
longip=Long.parseLong(ipAddressInArray[3- i]);
        //left shifting 24,16,8,0 and bitwise OR
        //1. 192 << 24
        //1. 168 << 16
        //1. 1   << 8
        //1. 2   << 0
    result |= ip << (i * 8);
    }
    return result;
     }
```

156 00000000 00000000 00000000 10011100

---

156 << 24 10011100 00000000 00000000 00000000
Result 00000000 00000000 00000000 00000000
Result |= 10011100 00000000 00000000 00000000

124 00000000 00000000 00000000 01111100

---

124 << 16 00000000 01111100 00000000 00000000
Result 10011100 00000000 00000000 00000000
Result |= 10011100 01111100 00000000 00000000

2 00000000 00000000 00000000 00000010

---

2 << 8 00000000 00000000 00000010 00000000
Result 10011100 01111100 00000000 00000000
Result |= 10011100 01111100 00000010 00000000

1 00000000 00000000 00000000 00000001

1<< 0 00000000 00000000 00000000 00000001
Result 10011100 01111100 00000010 00000000
Result |= 10011100 01111100 00000010 00000001

Convert the final binary code to integer.
Result 10011100 01111100 00000010 00000001
Index 0 - 31, start from right.
31(1),30,29,28(1),27(1),26(1),25,24,23,22(1),21(1),20(1),19(1),18(1),17,16,15,14,13,12,11,10,9(1),8,7,6,5,4,3,2,1,0(1)
Decimal 1x2^31+1x2^28+1x2^27+1x2^26+1x2^22 + 1x2^21+1x2^20+1x2^19+1x2^18+1x2^9+1x2^0
=

2147483648+268435456+134217728+67108864+4194304+2097152+1048576+524288+262144+512+1

2625372673

Table 3: IP address and equivalent integer value.

IP address	Integer Value
115.178.60.136	1941060744
10.35.22.72	170071733
71.45.142.70	1194167878
10.15.164.181	168797256
116.144.60.136	1955609736
94.135.64.187	158592235
115.146.25.168	1938954664
122.35.14.36	2049117732

25.13.45.126	420294014
145.23.65.84	2434220372
184.16.23.177	3088062385
158.56.15.156	2654474140
86.14.28.119	1443765367
145.36.58.79	2435070543
89.145.46.17	1502686737

After getting the integer value we normalized the values between 0 to 1

Normalization by Scaling

Assume that there are n rows with several variables, A, B, C, D, E, F, and so on in the data. We use variable X as an example in the calculations below. The remaining variables in the rows are normalized in the same way.

The normalized value of x_i for variable X in the i^{th} row is calculated as:

Normalized $(x_i) = (x_i - X_{min}) / (X_{max} - X_{min})$

Where

X_{min} = the minimum value for variable X

X_{max} = the maximum value for variable X

Table 4: Normalized inbound traffic.

TRAFFIC VALUE	TYPE
0.2517893	1
0.3578615	0
0.3458791	0
0 .4237801	1
0.5368745	1
0.2578350	0
0.4578612	0
0.597368	1
0.6491378	0
0.4618735	1
0.7845320	1
0.3571596	0
0.4957325	0
0.6741386	1
0.3547369	0

Only time was an integer and rest of the attributes should be converted into integer. So we calculated the entropy of the source ip address and destination ip address. After processing, the dataset contains 115928 samples. 70% of the data set was used for training and 30% of the dataset was used for testing.

Result and Discussion

We tried using K-means clustering on the dataset. After fine tuning the parameters of K-means we were able to get a best accuracy of 0.54. Here 2 clusters were used as dataset contained 2 categories of traffic attack and normal. The attack traffic was the traffic recorded when the network is attacked and the normal was traffic recorded the rest of the time. We tried using more than 2 clusters but we got the best results when using 2 clusters.

Figure 4: Graph for clustering dataset with 10 samples

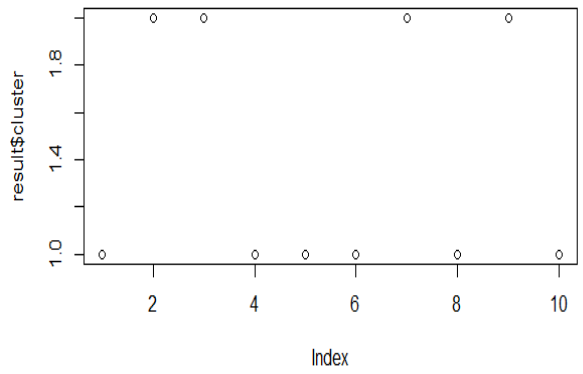


Figure 5: Graph for clustering dataset s with 50 samples

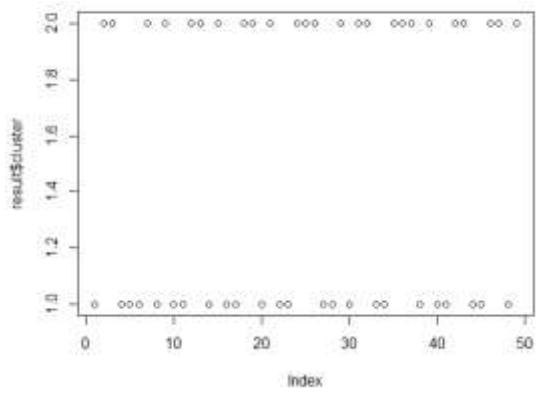


Figure 6: Graph for clustering dataset with 100 samples.

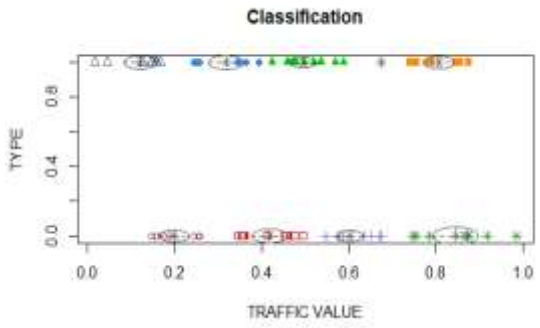


Figure 4, Figure 5 and Figure 6 shows clustering of datasets with 10, 50 and 100 samples respectively.

Deploying this in real time traffic monitoring applications shown in Figure 7. The time it takes to predict each label is crucial. The results got an average 0.00886 sec when used in real time environment for the best accuracy is shown in Figure 8.

Figure 7: Real time traffic monitoring application.

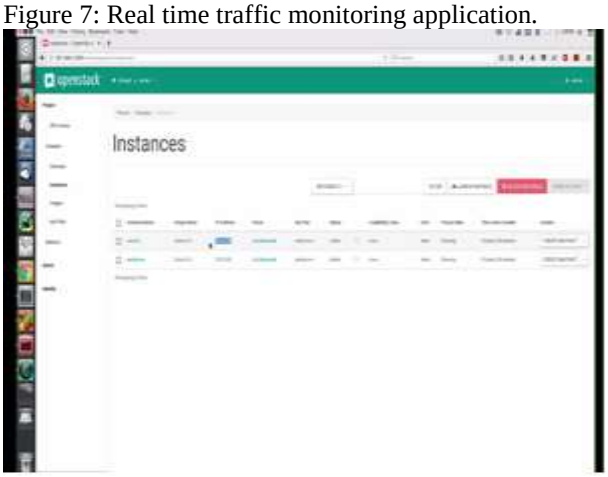


Figure 8: Finding the intruder in real time.

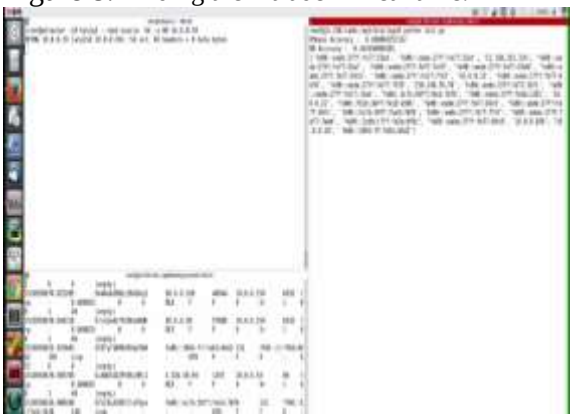


Figure 9: Comparison of Accuracy obtained by KMeans with other Algorithms

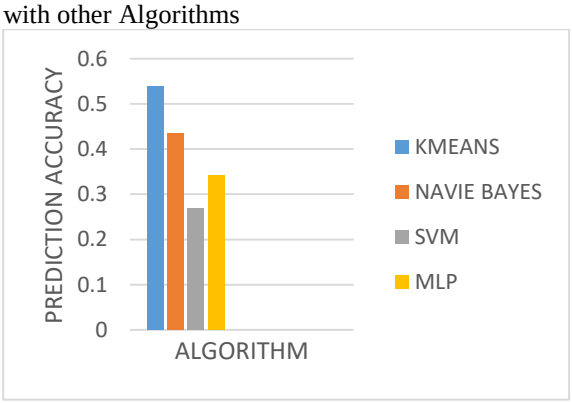
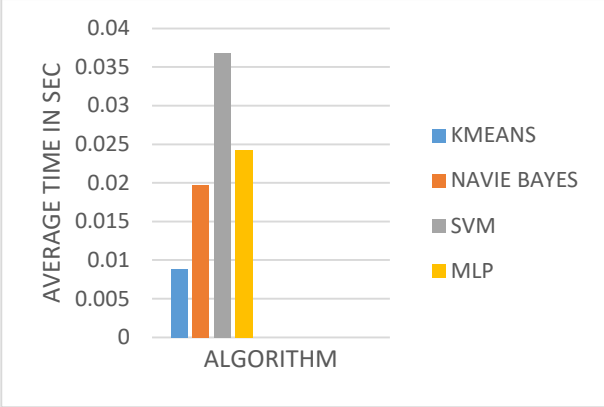


Figure 10: Comparison of average time taken to predict attacks by K-means and Navies Bayes Algorithm.



With Gaussian Naive Bayes we got an accuracy of 0.435 and an average time to predict of 0.0197 sec which is shown in Figure 9 and Figure 10 respectively.

Conclusion

In this work, we analyzed the techniques for anomaly detection in literature and proposed the solution for improve the performance of anomaly detection for modern day attacks. Finally, evaluated the proposed solution in a real time testbed. The datasets was collected from Canadian university to train the algorithm. K-means were able to get a best accuracy of 0.534 and got an average 0.00886 sec when used in real time environment. The system can be implemented on a variety of platforms, has a very simple approach, is easy to configure, does not incur any cost for implementation (as both routers and computer are already there in the network) and does not need any specialized person for its operation.

References

[1]Slaviero M, Black H at presentation demovids:Amazon, /http://www.sensepost. com/blog/3797.htmlS; 2009.

[2]Megiba R Jasmine, G. M. Nishibha,"Public Cloud Secure Group Sharing and Accessing in Cloud Computing",Indian Journal of Science and Technology,2015 July, 8(15), Doi no:10.17485/ijst/2015/v8i15/75177

[3]Stiawan D, Abdullah, AH, Idris, MY. "The trends of intrusion prevention system network". In: Second international conference on education technology and computer (ICETC) 4; 2010: 217–21.

[4]Roschke S, Feng C, Meinel C. "An extensible and virtualization compatible IDS management architecture". In: Fifth international conference on information assurance and security, 2; 2009: pp. 130–4.

[5]Dutkevyach T, Piskozub A, Tymoshyk, N. "Real-time intrusion prevention and anomaly analyze system for corporate networks". In: Fourth IEEE workshop on intelligent data acquisition and advanced computing systems: technology and applications, 2007. IDAACS 2007: 2007: pp. 599–602.

[6]Cox P. "Intrusion detection in a cloud computing environment". /http://searchcloud computing.techtarget.com/tip/Intrusion-detection-in-a-cloud-computing-environment; 2011.

[7]H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusiondetection systems," COMPUTER NETWORKS, vol. 31, no. 8, 1999.

[8] S. Bellovin and R. Bush, "Configuration management and security," IEEE Journal on Selected Areas in Communications fSAC, vol. 27, no. 3, 2009.

[9]L. Schaelicke, T. Slabach, B. Moore, and C. Freeland, "Characterizing the Performance of Network Intrusion Detection Sensors," Recent Advances in Intrusion Detection: 6th International Symposium, RAID 2003, Pittsburgh, PA, Usa, September 8-10, 2003.

[10]V. M. Igure and R. D. Williams, "Taxonomies of Attacks and Vulnerabilities in Computer Systems," IEEE Commun. Surveys Tutorials, vol. 10, issue 1, pp. 6-19, First Quarter 2008

[11]R. Smith, C. Estan and S. Jha, "Backtracking Algorithmic Complexity Attacks against a NIDS," In Proc. 22nd Annual Computer Security Applications Conference (ACSAC), Dec. 2006

[12]S. Dharmapurikar and V. Paxson, "Robust TCP Stream Reassembly In the Presence of Adversaries," In Proc. USENIX Security Symposium, Aug. 2005.

[13]U. Shankar and V. Paxson. "Active Mapping: Resisting NIDS Evasion without Alerting Traffic," In Proc. IEEE Symposium on Security and Privacy, May 2003.

[14]G. Taleck, "Ambiguity Resolution via Passive OS Fingerprinting," in Proc. International Conference on Recent Advances in Intrusion Detection (RAID), Sept. 2003.

[15]M. Sachs and D. Ahmad, "Always the Same, Never the Same," IEEE Security & Privacy, vol. 8, issue 2, pp. 73-75, Mar./Apr. 2010

