Journal of Engineering and Applied Sciences 13 (22): 9789-9799, 2018
ISSN: 1816-949X

Implementing Secure Cluster using Hadoop and Snort for ID (Intrusion Detection)

[1]Rafah M. Almuttairi, [2]Maad Kamal Al-Anni and [1]Dalya A. Aljburi
[1]Ministry of Higher Education and Scientific Research, Department of Information Networks,
College of Information Technology, University of Babylon, Hillah, Iraq
[2]Ministry of Higher Education and Scientific Research, Software and Networking Engineering College,
Al-Iraqia University, Baghdad, Iraq

Abstract: Among the best ways for someone or a company to get famous and well known is through certain electronic mediausing web applications to make customers know what companies have through their websites, as the formers shop and inquire through web services of the latters. Therefore, it is necessary to protect the website against any attack by those who are not interested in the progress of high-level companies and the spread of their fame. One of the attacks that the Web server is likely to experience is the Denial of Service Distributed (DDoS), across the application layer. The increased volume of data resulting from the attack makes the current detection systems inefficient to detect the hacker. In this research, a new methodology is proposed to detect and prevent attacks through the use of Hadoob framework which will accelerate the analysis of data to discover the attack and deliver it to the Snort to be blocked and stop harm. After the analysis of the data we found that the proposed could provide a 99.01% reduction rate, 99.27, 99.72% for the original alerts 1668, 2182 and 2698, respectively, compared to the traditional model.

Key words: Web server, Apache Hadoop, DoS, MapReduce, snort, DDoS, big data, alerts

INTRODUCTION

Cyber attacks have turned into an unavoidable truth. The most commonattack of cyber threat is the Denial of Service (DoS), that makes web servers and most online resources and web applications unavailable for intended web client and user agent. DoS attacks have three general classes: first is the attack to the seventh layer "Application layer", second is theattack to the forth layer " Transport layer" and thirdis the attack to the third layer "Network layer" of Open Systems Interconnection (OSI) Model. The Types of attacks could be either Denial of Service (DoS) or Distributed Denial of Service (DDoS) attacks. The threat seeks to overload a server by sending a large amount of requests which require a lot of processing. At the point when the attack is accomplished by just a solitary framework, it is been alluded as a Denial of Service (DoS) attack (Wang *et al*., 2015). While in the event that the attack is accomplished by various far off territories to accomplish a similar objective from various areas for this research, it is called as a Distributed Denial of Service (DDoS) a attack (McGregory, 2013). The attacksto the seventh, fourth and third OSI layers may consume Bandwidth of the web server that iscaused by manipulating with OSI layers protocols of web server, such as the seventh layer protocols, "Hyper Text Transfer Protocol (HTTP)" and "Domain Name Service (DNS)" and the third layer protocol, "Internet Control Message Protocol (ICMP)" and "Synchronize Sequence Numbers (SYN)" and the third layer protocols, "Transport Control Protocol (TCP)" and "User Datagram Protocol (UDP)"(Hunter, 2003).

Distributed Denial of Service (DDoS) attacks to the Application layer are usually hard to be detected and handled because this type of attacks can easily pass with normal packets via. the system of Intrusion Detection (IDS) or System of Intrusion Prevention (IPS). This type of attack uses genuine protocols trying to exhaust resources of web server or its network Bandwidth (Sood *et al*., 2013).

Network infrastructure and communications over Internet should be safe and secure from all previous defined attacks using secure applications such as Firewall. Firewall looks outwardly as a wall can be used for firing all attacks. It limits and prevents bad requests from continuing intrusion and accessing a Fig. 1. Percentage of DDoS attacks (25) any service of web server (Anonymouse, 2013). In the last few years, intrusion

Corresponding Author: Rafah M. Almuttairi, Ministry of Higher Education and Scientific Research, Department of Information Networks, College of Information Technology, University of Babylon, Hillah, Iraq

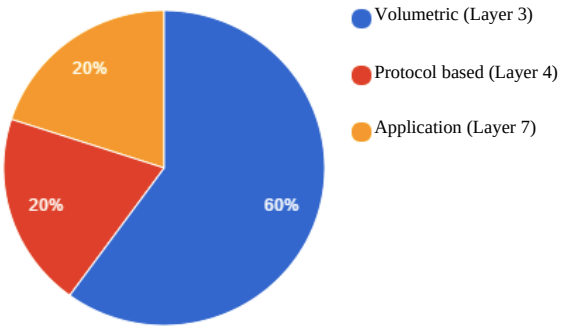

Fig. 1: DDos attacks

detection techniques has been improved using smart detection methods. However, the most effecient intrusion detection systems used are: Snort, Bro IDS, security onion, suricata and open-WIPS-NG, etc. Table 1 illustrates some of systems of intrusion detection. Snort can be considered as a very powerful Intrusion Detection and Preventation System (IDPS) where the suspected intrusion could be detected as soon as it has taken place. Snort tools sends negative alarm signals to administrator of network (Ismail and Taha, 2009). Snort sniffing all new packets and matching these signatures with all patterns which it has been stored inside it.

As shown in Fig. 1, attack on seventh layer of OSI Model which is called "Application Layer Attack" has pointing out a higher percentage of attacks as the recent statistic report published by Lease web knowledge base (McGregory, 2013).

The quick increment of distributed deniel of service attacks on the Internet brings up the restrictions of the conventional current Intrusion Detection System IDS or Intrusion Prevention System (IPS). The speed of current processing systems is not efficient enough to deal with huge and complex data sets (Somani *et al*, 2017), consequently the rapid increase of DDoS attacks on the Internet has clearly pointed out the limitations in current intrusion detection systems IDS or intrusion prevention systems IPS, Big data is a term related to growing data, data might be unstructured, structured or semi structured that has the potential to be quarried to get required information.

The most big data softwares used are illustrated in Table 2. The researchers concentrate on malicious attacks. In case the available network security techniques do not updated to cover their limitations to research with huge datasets, then network hacking crime is going to be increased.

To improve the capability of Snort, the current study is proposed which is an integrated system that includes Snort application with Hadoop infrastructure. Therefore, the limitation of dealing with huge data setes could be covered. A new integrated system is scalable. It can increase the ability of Snort by dealing with production mode of Hadoop where multiple nodes will be running. Here data is distributed across several nodes and processing will be done on each node. Master and Slave services will be running on the separate nodes in fully-distributed Hadoop mode. Finally, we can say thatthe total processing time is going to be shorter whenever the datasets is increasing and that is exactly what our experiments have proved.

Literature review: A network intrusion detection system framework based on Hadoop and GPGPU (Bandre and Nandimath, 2014) in this study they attempt to design an a NIDS system for handling large scale data on a network by using parallel computing CUDA technology and Hadoop data-platform. The goal of this approach is to optimize NIDS overall performance through offloading intrusion mapping tread-off functionality from Hadoop to GP-GPU. The scientists have configured NIDS with Hadoop information platform so that it can be tackled and sustained big volumes of incoming traffic. They discovered that the approach is capable of processing log dataset of 1, 2 and 4 GB in a total potential time that of 29.86, 47.09 and 94.96 sec. The approach brings about optimizing NIDS overall performance.

Quantitative evaluation of intrusion detection model: Snort and Suricata in this study, they contrast among Suricata and snort tools. Suricata is a multi-threaded substitution to snort construct In 2009 by means of OISF enterprise. They discover that a unique example of Suricata has higher overall performance than a unique example of snorting, however Suricata has scaling issues while there's an excessive range of cores.

Survey on signature primarily based intrusion detection approach by the usage of multi-threading (Roka and Naik, 2017) in this study, they offer a brand new technique for a multi-threaded IDS to enhance thesignature primarily based IDS overall performance to enhance the stand-alone CPU overall performance restriction. They evolved a multi-threaded NIDS sensor. This permits to unfold the signature over more than one CPU of the same machine and overall performance to scale post stand-alone-CPU machines.

A suggestion for implementation of signature is primarily based intrusion detection approach by the use of multithreading method (Gaikwad *et al*., 2012) in this study a signature primarily based IDS, the usage of

Table 1: Cons and pros of network based intrusion detection system

IDS	Pros	Cons
Snort	It is easy to install, get up and running online resource is support available online	Contains GUI add-ons exist Packet processing is slow
Suricata	Snort's rulesets can be used propelled features such as multithreading abilities acceleration of GPU	Likelyto positives that are not true Intensive work and resource of network
Bro IDS	Suitable place for multiple network security situations and to NIDS	The experience of programming is necessary The proficiently skills in Bro DSL need some efforts
OpenWIPS-ng	It is Modular and Plug-in the software and hardwareare built using "DIYer"	Mainly used for a wireless security solution
Security-Onion	It is a security stack Provides an easy setup tool leads to an open-source solutions	It is aplace that is made up of different l technologies itgets the drawbacks of the constituent tool

Table 2: Hadoop vs. tez vs. spark

Criteria			
License	It is an open source "apache server x Server 2.0,Ver. .x	Also open source Apache server 2.0 Ver. 1.x	Also open source Apache server 2.0 Ver. 1.x
Models used for processing	1-On-Disk (Disk-based parallelization),2-Batch	1-On-Disk2-Batch 3-Interactive	1-In-Memory 2-On-Disk3-Batch 4-Interactive, 5-Streaming (near real-time)
Written language is	Java	Java	Scala
API	1-Java 2-Python3-Scala 4-User-interfacing	1-Java, 2-ISV/Engine/tool builder	1-Scala 2-Java 3-Paython 4-User facing
Libraries	Separate tools	None	1-Spark Core, 2-Spark Streaming3-Spark SQL 4-Mlib, 5-GraphX

multi-threading approach is recommended. The network request packet can be treated via. a multi-threading approach. They recommend packet flow capture module for capturing time-being packets from the network and a module for multithreaded layout. This research takes the cons of parallel processing on captured packets the following is DDoS detection technique primarily based on a threshold for HTTP GET request. The HTTP Get Flooding attack is the most important and regularly attempted attacks and the threshold is generated from the attributes of HTTP GET Request behaviors (Anonymouse, 2013). DDoS detection approach primarily based on a threshold for HTTP GET Request is brief of accurateness due to the fact that the threshold is certain to be excessive. specifically, the traditional technique is vulnerable to up-to this point DDoS attack that paralyzes the system with smallquantities of HTTP requests in the past, Layer 4 treated as a major goal for depletion of connection assets between network devices. But, internet applications turned into a major goal lately due to the fact DDoS corresponding solutions were correctly replying to the prevailing DDoS attacks. Software level DOS attacks emulate the identical request Syntax and network level traffic requests characteristics as those of valid parties, thereby making the attacks lots more difficult to discover and counter. Software-level attacks are HTTP GET Flooding, refresh attack, square Injection attack, CC attack and so forth (Anonymouse, 2010; Suriadi *et al*., 2011; Bakshi and Dujodwala, 2010) ultimately, detection technique primarily based on scalability is a technique for resolving drawback between the above techniques. The main study contents of this technique are a categorical approach through making use of statistical parametric from the dataset, a technique making use of Bigdata paradigm through networking pattern evaluation, a detection technique of falsification HTTP message attack via. tracking HTTP request and so forth.

Important concepts: In this study, we will review some of the important concepts and topics that relate to our study.

MATERIALS AND METHODS

HTTP GET flooding attack: An HTTP flood is an attack approach utilized by hackers to attack internet servers and assets. These illegitimate HTTP GET or http requests are particularly designed to deplete a substantial quantity of the server's assets (with outrequiring an excessive portion of network traffic). It is very difficult for network protection devices to differentiate between valid HTTP traffic and malicious HTTP traffic and if no longer treated successfully, it is able to motivate certain excessive quantity of false-positive detections. Rate-based detection engines are also no longer successful at detecting http flood attack. as the traffic volume of HTTP floods may be below detection thresholds. Because of this, it is necessary to use several parameters detection

Table 3: Configration of attack tools

Tool name	Operating system	Memory (GB)	Processors	Hard disk (GB)	Network adapter	No. per OS
Slowhttptest	Red Hat 6.6	2	1	800	Bridged	1
LOIC	Win8	1	1	500	NIC	4
HOIC	Win7	1	1	500	NIC	3
RUDY	CentOS 6.8	2	1	800	Bridged	6
BONESI	Fedora 24	1	1	800	bridged	1
Anonymous DDoSer	Win 7	1	1	500	NIC	5
HAVIJ	Win 8	1	1	500	NIC	5

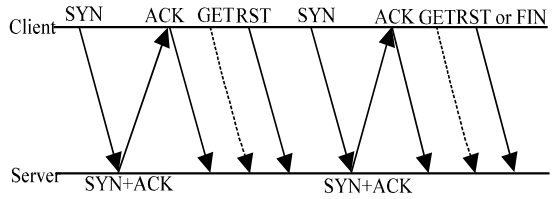

Fig. 2: Packet flow of HTTP GET request attack

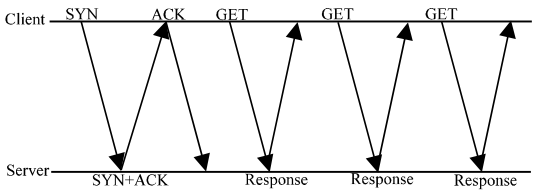

Fig. 3: Multiple HTTP GET request in single TCP connection

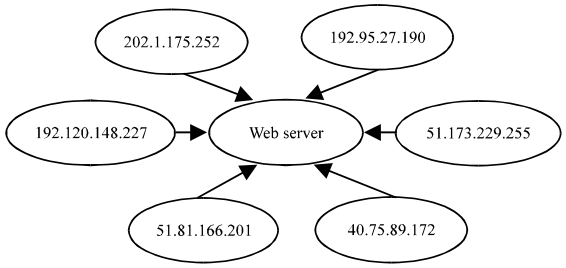

Fig. 4: Typical DDoS attack scenario

including rate-based and rate-invariant (Choi *et al*., 2012). Figure 2 illustrates the packet flow of HTTP GET request attack after single TCP connection.

Intensively multiple HTTP GET request in TCP connection are dispensed HTTP attack template using httpd service breaches as illustrated in Fig. 3.

This technique is for giving a heavy load to the identical web page or the database server via more than one HTTP GET request in a single connection.

Attack scenario: A layer 7 DDoS attack on an internet server is achieved via more than one abnormal attacker sending huge HTTP GET requests to the HTTP server. but in the case of an attack, the attackers are set up with the automatic program (with information given in Table 3) and consequently, transmit an HTTP GET request in a specific fashion to crash the http server. Fig. 4 provides a standard DDoS attack situation.

In this research, the randomness of the flow is shown with the aid of computation of entropy as we are going to reveal it in coming sections. A smaller rate of entropy will exhibit that the flow is in order in any other case the flow is out of order. In the case of an attack, the mean entropy rate will be small while in comparison with a daily normal scenario. We take into account the IP addresses 51.81.166.201 and 192.95.27.190 in Fig. 4, as a normal HTTP request, even as the rest are attacking http request. In Fig. 4, hereafter, we display only six IP addresses for easier and faster statistical analysis of the attack and the normal daily http scenario. While the entire number of clients taken into consideration for gaining access to the net server is 357. Each attack attacker is mounted with an automatic software to launch a big HTTP GET request. We are going to expose the experimental setup, examine the HTTP GET request, Entropy and the variance for every IP connection at specific instants of time.

Transmission Control Protocol (TCP): The Transmission Control Protocol (TCP) is an important protocols of the TCP/IP suite. TCP presents a dependable information flow. TCP is a well-known transport layer used with internet protocol. TCP makes use of headers as partition of packaging message information to hold over network connections. TCP headers comprise of units of fields as follows (Fig. 5).

Every TCP header has 10 required fields there general length is 20 bytes (160 bits). They also can consist of an option all extra recordsegment, it's length as much as 40 bytes (Almuttairi *et al*., 2010). We analyzed the data set generated by using a live DDoS attack acquired via. experimental setup and widespread EPA-HTTP (Environmental Safety Agency-hypertext Transfer Protocol) datasets (Anonymouse, 2015).

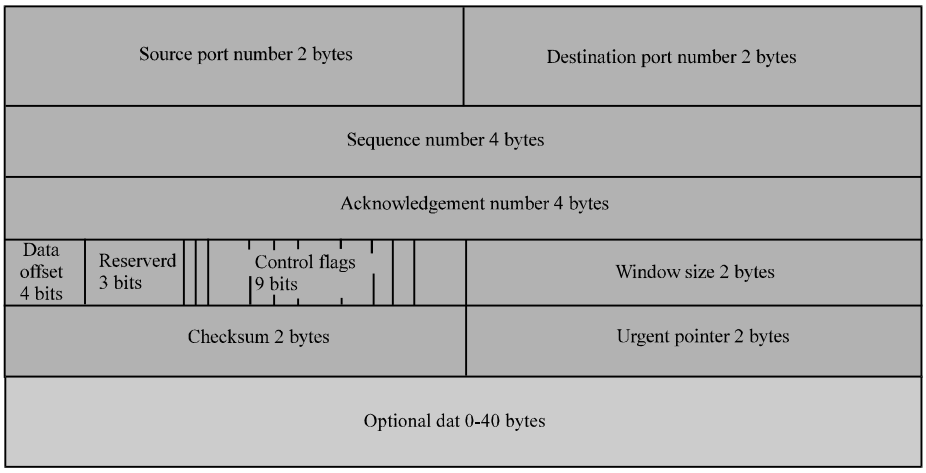

Fig. 5: Transmission Control Protocol (TCP) header

HTTP GET request count: HTTP is an essential software-layer protocol used for hypertext document exchanges (Jestratjew and Kwiecien, 2013). There are numerous strategies in HTTP which include get, post, put, delete, head, connect, trace and options that are used for thetele-conversationin this context, we take into account the HTTP GET request. We rely the range of HTTP GET requests for a specific connection within a time window of 20 sec. We take into account 20sec used as the time window due to the fact most of the automatic equipment's (like slowloris, High Orbit Ion Cannon (HOIC) (Hoque *et al*., 2014), Low Orbit Ion Cannon (LOIC) (Hoque *et al*., 2014) and so on) is programmed to bring down an internet server within this interval.

We consider the number of HTTP GET requests for a specific IP address through making a time slot of 20 sec. The evaluation is made feasible through putting in a sniffing tool (Hoque *et al*., 2014) at the focused web server. Following algorithm HTTP GET flow count offers a method for counting the number of HTTP GET requests, on this context, this type of http get request depend on managing by using both conventional technique or scalable strategies, then the assessment of comparison is additionally being mentioned.

Algorithm 1; HTTP get flow count (GET requests) for the N participating clients for every 20 sec time window:

1: Begin
2: for (Frame.Time$_i$ = strx;Frame.Time$_i$<Frame.Time$_i$+20;i++)
3: for (Ip$_j$ = 1;IP$_j$ = N;j++)
4: ComputeI = (Ip$_j$ and IP$_{dst}$)
5: Compute H$_{GET}$ = (http.request.method within 20 sec == GET)
6: Compute Final(for 6 IP Address) = I and H$_{GET}$
7: endfor
8: end for
9: end

Algorithm 2; Term explenations:

Frame time$_i$	indicates the i-th time at which the packet capture Starts
STRXD	enote the start time
IP$_I$	denotes the j-th IP address of the participating client
IP$_{DST}$	refers to the IP address of the victim server
H$_{GET}$	indicates the selection of only the HTTP GET request
Final	total flow count of HTTP GET requests for a j-th client during a 20 sec time window

Our HTTP server is an Apache Wamp server (64 bits and PHP 5.5) 2.5 hosted on Windows running machine 7 (OS) with 8 GB RAM, 1 Terabit secondary storage and Intel 7 processor. The attack equipment had been prepared at various OS which includes Cent OS 6.8, Ubuntu 12.04, Fedora 24, Home Windows 7 and Window 8. To increase the number of attacker OS in addition to valid clients, we hooked up at the least 4 OS with the assist of VMware workstation model 10.0.3 in every machine. We additionally ran at the least 4 attack equipment's in every OS.

To capture all of the incoming packets from various clients in the direction of the focused server, we hooked up a Sniffing device (snort) of model 1.12.6 onUbunu 14. xOS website hosting the Wamp server. snort as a packet analyzer can capture packets moving into and out of the closed surroundings by using adjusting the Network Interface Card (NIC). Consequently, we could get experimental log files of the DDoS attack situation that even contained valid client's packets.

Table 4-8 display the captured tcp/ip flow be counted throughout the first time slot window as much as the 5th window, respectively. It additionally indicates the tcp connection between source IP address and the target IP address. For this study, we picked up the exceptional six connections for demonstration.

Table 4: Flow count for the first 20 sec time window

Source address	Destination address	The count
192.168.4.102	192.168.5.113	1213
192.168.4.103	192.168.5.113	1242
192.168.4.104	192.168.5.113	2210
192.168.4.105	192.168.5.113	1856
192.168.4.106	192.168.5.113	7390
192.168.4.107	192.168.5.113	1182

Table 5: Flow count for the second 20 sec time window

Source address	Destination address	The count
192.168.4.102	192.168.5.113	6190
192.168.4.103	192.168.5.113	6370
192.168.4.104	192.168.5.113	7800
192.168.4.105	192.168.5.113	1025
192.168.4.106	192.168.5.113	3220
192.168.4.107	192.168.5.113	6000

Table 6: Flow count for the third 20 sec time window

Source address	Destination address	The count
192.168.4.102	192.168.5.113	1229
92.168.4.103	192.168.5.113	1278
192.168.4.104	192.168.5.113	3010
192.168.4.105	192.168.5.113	1803
192.168.4.106	192.168.5.113	2170
192.168.4.106	192.168.5.113	1200

Table 7: flow count for the fourth 20 sec time window

Source Address	Destination address	The count
192.168.4.102	192.168.5.113	1253
192.168.4.103	192.168.5.113	1273
192.168.4.104	192.168.5.113	3340
192.168.4.105	192.168.5.113	1734
192.168.4.106	192.168.5.113	7700
192.168.4.107	192.168.5.113	1099

Table 8: flow count for the fifth 20 sec time window

Source address	Destination address	The count
192.168.4.102	192.168.5.113	1236
192.168.4.103	192.168.5.113	1277
192.168.4.104	192.168.5.113	3770
192.168.4.105	192.168.5.113	1713
192.168.4.106	192.168.5.113	1160
192.168.4.107	192.168.5.113	1148

Snort: Snort is a free and open source Tools use for Network Intrusion Detection System (NIDS) and Network Intrusion Prevention System (NIPS). It laisses primarily based logging the capture dataset to match attack signatures within the predifined stes of rules and discover a variety of attacks and probes through diverse alert messages, those attacks such asPhf attacks, Imap attacks, Sendmail guest attack and plenty more, Snort has real-time proactives either via sending it to SYSLOG, database, or a separate "alert" file (Roesch, 1999). Snort is a Signature-primarily based IDS/IPS, it works on predefined rules for content pattern matching toquest attacks, those rules can be downloaded through registered user and then to consist of set-up rules one to trade off the configuration file while rules are matched with the captured packets. Therefore, actions laid out in

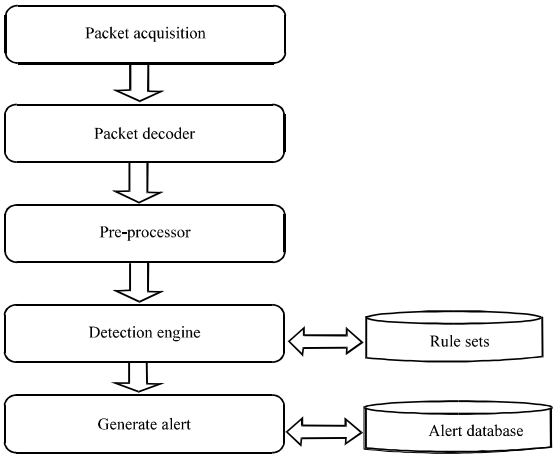

Fig. 6: Snort architecture

the rules could be fired, those actions can be of four assorts it could be to alert the nodes, log the captured packets, bypass the packet with none action or to drop the packet.

Snort structure contains of three subsystems as illustrated in Fig. 6 the packet decoder, detection engine and the logging and alerting subsystem. The decoder because the name indicates decodes the incoming packets to the network. decoding is referred tocarry out the function of decoding the incoming packets into other form bythe order via. protocol stack from the data link layer up to the application layer, hyperlink layer as much as the datalink layer. The detection engine contains the predifined detection rules in a two-dimensional related list. these rule chains are lookingfor each incoming packet within predifined rules in two-dimension. While a rule in the detection engine corresponds the captured packets, the action designated in the definition of a rule which will be triggered, consequently able to be either to alert or to log the packets.

An instance of a Snort rule is alert tcp any >any any (msg:"probability of an attack") which implies thet snort needs to warn any IP address if TCP packet originates from any source port or target port. while the rule is coorsponded both logging or alarming can be chosen through console commands. Within the logging subsystem, the packets could be logged to a subdirectory in a human readable file or in a tcpdump binary arrangement. Alarming subsystem send cautions to typical content report or a database like MySql or to Syslog.

Sniffing tool (Snort) has three distinct modes, sniffer mode, packet logger mode and network intrusion

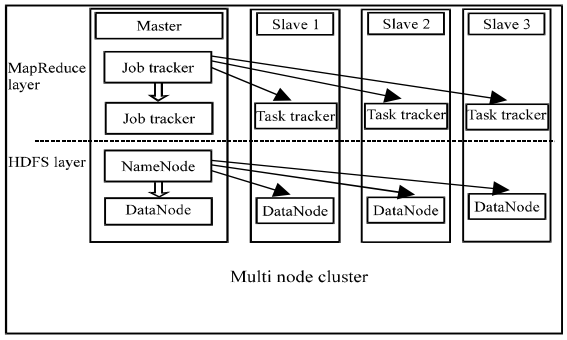

Fig. 7 Hadoop architecture

detection mode. Sniffer mode detects the incoming packets and shows them in console. In packet logger mode, Snort collects the packets and logs them to disk. Network Intrusion Detection System (NIDS) mode cautious actions might be produced in various techniques, Alerts also can be produced in any such manner that it will show best beneficial information or will show the relevant information for TCP/IP packet detials about attack senarios. It can be picked by suitable Snort orders. in this research, the approachis caught for cutting edge investigation by running the snort in packetlog mode in order to log the incomming packets.

Hadoop: Figure 7 suggests the Hadoop structure; Hadoop is essentially a framework for performing programs on BigData. It allows massive of nodes to execute together in large cluster for doing a one task activity. Hadoop is for manage peta bytes of records in an aggregately limited time. because of the linear relationship when the number of nodes are being increased, then the time taken to process such a dataset are decreased in such a way that shows the scalability performance with respect of time periodic. Hadoop is written in Java and user applications might be written in Java, Python and Ruby etc. The two important components of Hadoop are Distributed Storage and Distributed Processing. Distributed storage is namely given by Hadoop Distributed File System (HDFS) and Distributed Processing is done by known as MapReduce. Those are the spines of Hadoop structure. A simplest Hadoop clustering architecture will comprise of a solitary master nodes and numerous working nodes (slaves). The solitary master node comprises of a job tracker, task tracker, name node and data node. A slave or working node goes about as both an data node and task tracker. Job tracker and task trackers are in charge of doing the map reduce tasks. Name nodes and data nodes are the piece of the distributed Data File System (HDFS).

Hadoop Distributed File System (HDFS): HDFS is a Distributed, Versatile and transportable file system written in Java for the Hadoop structure. Files are separated into gigantic blocks and are dispersed all through the cluster. standard piece volume is 64 MB. The block sizes can be changed. To deal with fault tolerance if in case it is occurred, blocks might be recreated through the appropriate replication way. Each node in a Hadoop generally has a solitary DataNode. HDFS cluster is shaped by means of a group of Data Nodes. A Name Node plays out various operation upon file system which incorporates close, open, rename and so forth, Name Node moreover manipulate a replication of distributed blocks. Data nodes run operations comprising of read and write that require to handle the file system operation at the client side. A file is split out into more than a block that is put away in an DataNode and HDFS decides the mapping between DataNodes and blocks (Chen *et al*., 2009).

MapReduce: MapReduce is a framework for running applications on BigData which have critical huge volumes of dataset in parallel paradigm spreading on enormous clusters having a huge number of nodes in a reliable and fault tolerance style. The MapReduce process that is situated over the file system comprise of one job tracker. Working nodes present MapReduce jobs to this job tracker. The Job tracker looking for the available task tracker to push up the workload. amongst the most significant feature for Hadoop that is rack-mindful. With rack-mindful file system framework, the job tracker knows about which node comprises of the data and which other machines are close by. In the event that a node breaks down or no longer, then the task is given to nodes in a similar rack. This mechanism can keep the system up and reduce the network traffic.

In Fig. 8, Map and Reduce are two central highlights inside the MapReduce computation. The map's idea is to break the input file that is a big one into smaller ones. Suitable tasks are properly done on those broken blocks or smaller ones of the big file that Reducer's merge these parallel processed data and produce a single output. The Hadoop MapReduce library will unite those intermediate values relating to a similar key whereas the main function of MapReduce is to have the mapping according to the certain key and later on doing the merge task, finally it is more beneficial for redundancy method of original file.

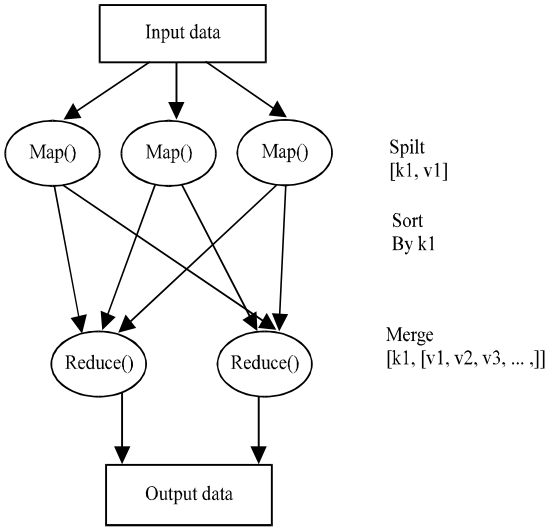

Fig. 8: MapReduce

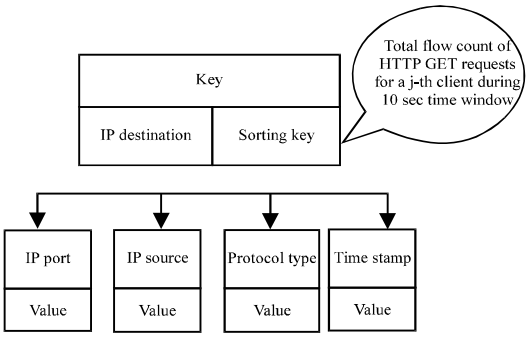

Fig. 9: Key/value pairs for HTTP GET flooding attack

Those calculation takes A set of enter key/value pairs to HTTP GET flooding attack, also produces a set of out-put key/value pairs as illustrated on Fig. 9.

Entropy and variance of the connection: For every specific connection, we ascertain the entropy rate of the HTTP GET flow count given through Table 4-8. The entropy E(i, w_i) for a specific connection C (I, w_i) within those time slotw_i is provided in Eq. 1 (David and Thomas, 2015). The calculated entropy rates for those connections need are given in Table 9:

$$E(i,w,i)=-\log\frac{C(i,w,i)}{\sum_{i=2}^{n}C(i,w,i)}+\lambda(i,w,i) \quad (1)$$

Where:

Table 9: Enteropy, mean, Variance and approximate for different time windows

	Entropy				
Source address	1st window	2nd window	3rd window	4th window	5th window
192.168.4.102	1.018	1.022	0.544	0.669	0.678
192.168.4.103	1.006	1.014	0.521	0.657	0.663
192.168.4.104	1.918	2.211	1.192	1.290	1.319
192.168.4.105	0.799	0.751	0.387	0.527	0.548
192.168.4.106	1.302	1.179	1.739	2.052	2.106
192.168.4.107	1.032	1.038	0.585	0.738	0.745

$$\left|\log\frac{^{C}(i,w_{i+1})}{^{C}(i,wi)}\right|,\ ^{C}(i,w_i)\geq{}^{C}(i,w_{i+1})$$

$$\lambda(i,w_i)=$$

$$\left|\log\frac{^{C}(i,w_{i+1})}{^{C}(i,wi)}\right|,\ ^{C}(i,w_i)<{}^{C}(i,w_{i+1})$$

We expect the variance of the entropy rate for the reason that calculating the variance value exhibits the varitions inside the entropy value. The variance V for a chosen connection C_i is made through Eq. 2 wherein M is the mean rate of the entropy and N is the number of entropy rate is considered. The calculating values of the variance for the particular connection are given in Table 9. In the event of an attack , there might be an about same style of packet transmission. Therefore, the variance cost is near zero when compared with the normal client situation:

$$V=\frac{\sum_{i}^{n}(c_i-M)^2}{N} \quad (2)$$

RESULTS AND DISCUSSION

We utilized EPA-HTTP datasets and lab datasets for analyzing both the proposedand traditional methods utilized with live DDoS datasets. From the assessment of the dataset done in Fig. 10-12, it is discovered that attack clients have a high flow account number but rate for entropy and variance is lower. Throughoutfalsh activities, the clients show higher flow count rate and higher rate from entropy rate due to disorder inside the request rate along with a higher variance rate because of the high variance rate inside the flow.

From Table 9, we discovered that theentropy rate of the 3rd and 5th IP addresses is higher than the rest. It's concluded that for a normal client, the level of randomness is rather bigger than that for the attack clients. This additionally indicates that human clients commonly do no longer have the capacity to post a similar

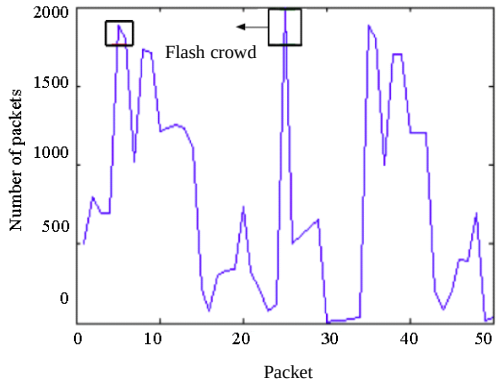

Fig.10: Http get request count for sniffing packet

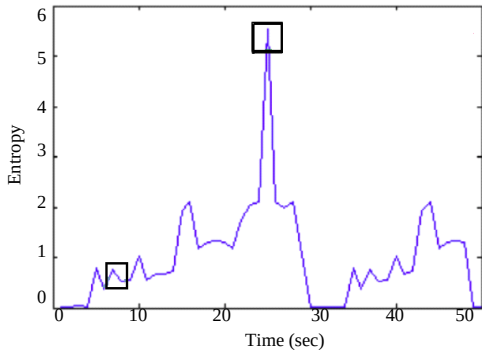

Fig. 11: Mean entropy for each particular IP connection

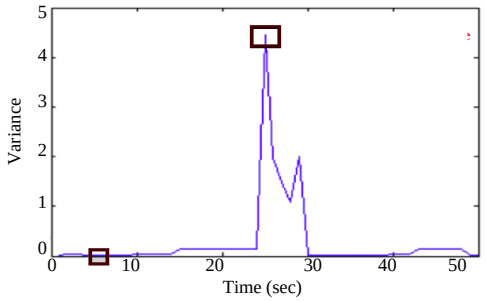

Fig. 12: Variance of the entropy for each particular IP connection

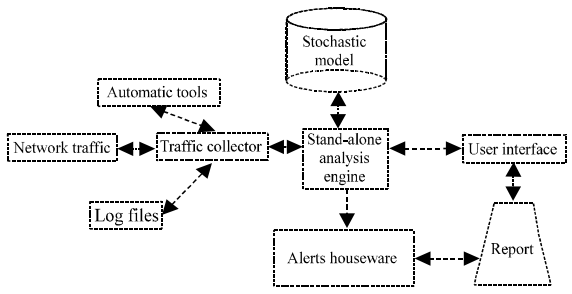

Fig. 13: Stand-alone analysis model

Table 10: Source, address, mean variance and approximaent for different time window

Source address	Mean	Variance	Approximate
192.168.4.102	0.786	0.0386874	0.039
192.168.4.103	0.772	0.0402854	0.040
192.168.4.104	**1.586**	**0.1629980**	**0.163**
192.168.4.105	0.602	0.0139102	0.014
192.168.4.106	**1.676**	**0.1434260**	**0.143**
192.168.4.107	0.828	0.0319508	0.032

*Bold values are significant

Table 11: Throughput data overall

	Analysis time (sec)						
Original alerts	Stand alone	1 Node	2 Nodes	4 Nodes	6 Nodes	Results	Reduction rate (%)
286	1.068	4.087	4.864	4.864	5.077	30	89.51
380	1.333	4.940	5.069	5.067	5.097	11	97.11
434	1.760	4.610	5.066	5.068	5.090	90	97.93
754	3.145	5.066	5.093	5.038	5.096	16	97.88
1174	4.730	6.066	5.093	5.089	5.097	33	97.19
1668	7.909	6.070	6.560	6.071	5.082	16	99.04
2182	14.949	6.671	6.950	5.166	5.088	16	99.27
3396	19.901	7.053	6.654	5.076	5.091	68	98.00
5816	374.374	9.081	9.076	9.070	7.076	66	98.87
6344	383.820	9.680	9.872	7.069	6.096	72	98.87
2698	801.346	13.096	12.367	11.367	9.083	36	99.72

number of HTTP GET requests. The range of http requests despatched for the duration of first 10 sec changed into different from the number of requests despatched afterward whereas an attacker has the capability of giving a similar range of the requests until the web server of the web turns into offline.

From Table 9, we additionally determined that the variance of the entropy for the attack on the IP addresses is less than that of the normal client's request. The variance rate for the attack clients throughout a hundred is near zero which leads to the realization that there's much less variation in the entropy rate. It's far different from that of the everyday client's request for which the variance of the entropy rate for the duration of 100 sec is huge. It is concluded that an everyday client has a huge variation in producing the HTTP GET request in this study illustrates two different approaches, the first one is primarily based on stand-alone paradigm as illustrated in Fig. 13 while the second one is extended to be a disbursed paradigm as illustrated in Fig. 14.

From Table 10 and 11, it is observed that we can achieve 99.72% reduction rate ((A-B)/A*100) in the distributed analysis model for original alert 2698 which is more than that of simply using a stand-alone model, It expresses that there is limited change in the generation of the number of HTTP GET requests through the period of the attack. The proposed approach may detect the DDoS attack with a reduced rate of 99.27, 99.04, 99.72% for original alerts 2182, 1668 and 0.9962 and 2698.

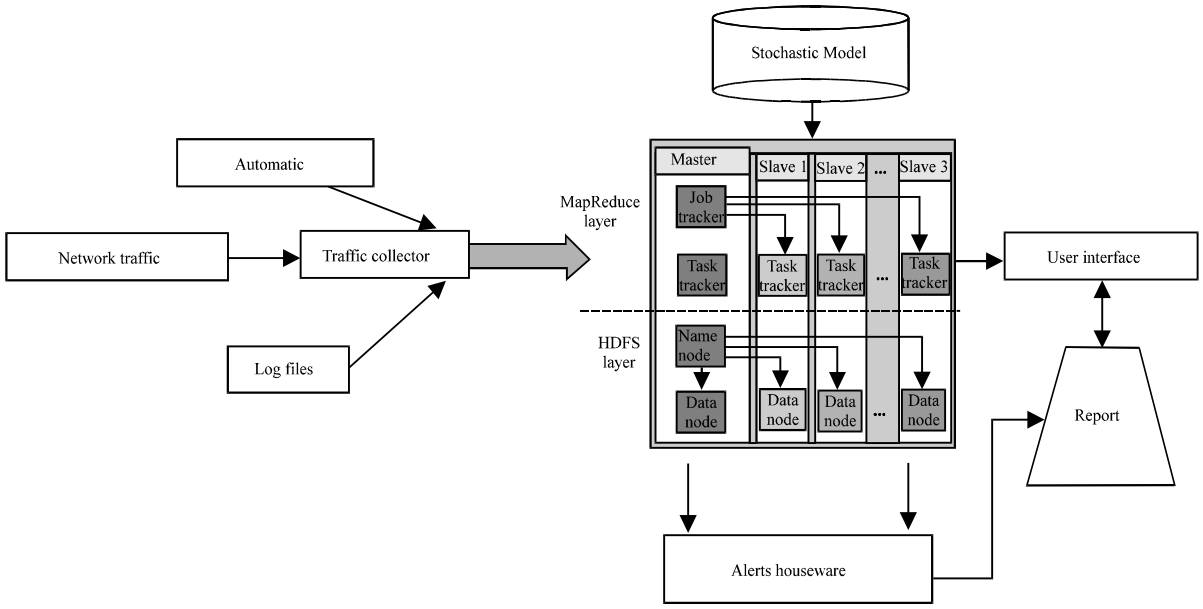

Fig. 14: Stand-alone analysis model

CONCLUSION

This study proposes a procedure of joining between HTTP GET flooding among DDOS attacks and MapReduce handling for a quick attack recognition in clustering environment. This procedure is conceivable to guarantee the avialability of the objective system for accurate and reliable detection principally built on HTTP GET flooding. In experiments, the processing time for performance evaluation analyzes of attack features with the proposed method (distributed analysis mode) is better than the traditional model (stand-alone analysis model in experiment outcomes due to the fact processing time of the proposed technique is shorter with increasing original alerts. Future research needs the look at of variants of DDoS attack detection in the cloud computing enviornment.

This research is confined best to application-layer DDoS attack identification. In any case, its aversion component isn't shown in the study.

SUGGESTIONS

In future research, we can augment the method to disclose the variant attacks on the httpd server with by imposing smart rules based on BigData paradigm to block the distinguished attacks or novel attacks. The method could be outfitted toward developing effective discovery for outstanding assortments of DDoS attacks.

REFERENCES

Almuttairi, R.M., R. Wankar, A. Negi and R.R. Chillarige, 2010. Rough set clustering approach to replica selection in data grids (RSCDG). Proceeding of the 10th International Conference on Intelligent Systems Design and Applications, Nov. 29-Dec. 1, Cairo, Egypt, pp: 1195-1200.

Anonymous, 2010. Study on the detection and mitigation algorithm for session consuming DDOS attacks on web service. Korea Internet & Security Agency, Seoul, South Korea.

Anonymous, 2013. DdoS attack definitions-DDoSPedia. Radware, Tel Aviv, Israel. https://security.radware. com/ddos-knowledge-center/ddospedia/http-flood/.

Anonymous, 2015. Advantages and disadvantages of firewalls computer science essay. UK Essays, Nottingham, UK. https://www.ukessays. com/essays /computer-science/advantages-and-disadvantages-of-firewalls- computer- science- essay.php.

Bakshi, A. and Y.B. Dujodwala, 2010. Securing cloud from DDOS attacks using intrusion detection system in virtual machine. Proceedings of the 2nd International Conference on Communication Software and Networks (ICCSN'10), February 26-28, 2010, IEEE, Singapore, ISBN:978-1- 4244-5726-7, pp: 260-264.

Bandre, S. and J. Nandimath, 2014. A network intrusion detection system framework based on Hadoop and GPGPU. Intl. J. Sci. Eng. Res., 3: 1-5.

Chen, W.Y., W.C. Kuo and Y.T. Wang, 2009. Building IDS log analysis system on novel grid computing architecture. National Center for High-Performance Computing, Taiwan.

Choi, Y.S., I.K. Kim, J.T. Oh and J.S. Jang, 2012. Aigg Threshold Based HTTP get Flooding Attack Detection. In: Information Security Applications, Lee, D.H. and M. Yung (Eds.). Springer, Berlin, Germany, ISBN:978-3-642-35415-1, pp: 270-284.

David, J. and C. Thomas, 2015. DDoS attack detection using fast entropy approach on flow-based network traffic. Procedia Comput. Sci., 50: 30-36.

Gaikwad, D.P., P. Pabshettiwar, P. Musale, P. Paranjape and A.S. Pawar, 2012. A proposal for implementation of signature based intrusion detection system using multithreading technique. Intl. J. Comput. Eng. Res., 2: 59-65.

Hoque, N., M.H. Bhuyan, R.C. Baishya, D.K. Bhattacharyya and J.K. Kalita, 2014. Network attacks: Taxonomy, tools and systems. J. Netw. Comput. Appl., 40: 307-324.

Hunter, P., 2003. Distributed Denial of Service (DDOS) mitigation tools. Netw. Secur., 2003: 12-14.

Ismail, M.N. and M. Taha, 2009. Framework of intrusion detection system via snort application on campus network environment. Proceedings of the IEEE, International Conference on Future Computer and Communication, April 3-5, 2009, Kuala Lumpar, pp: 455-459.

Jestratjew, A. and A. Kwiecien, 2013. Performance of HTTP protocol in networked control systems. IEEE. Trans. Ind. Inf., 9: 271-276.

McGregory, S., 2013. Preparing for the next DDoS attack. Netw. Secur., 2013: 5-6.

Roesch, M., 1999. Snort: Lightweight intrusion detection for networks. Proceedings of the LISA 13th International Conference on Systems Administration, Vol. 99, November 7-12, 1999, Stanford Telecommunications, Inc, Santa Clara, California, pp: 229-238.

Roka, S. and S. Naik, 2017. Survey on signature based intruction detection system using multithreading. Intl. J. Res. Granthaalayah, 5: 58-62.

Somani, G., M.S. Gaur, D. Sanghi, M. Conti and R. Buyya, 2017. DDoS attacks in cloud computing: Issues, taxonomy and future directions. Comput. Commun., 107: 30-48.

Sood, A.K., R.J. Enbody and R. Bansal, 2013. Dissecting SpyEye-understanding the design of third generation botnets. Comput. Netw., 57: 436-450.

Suriadi, S., D. Stebila, A. Clark and H. Liu, 2011. Defending web services against denial of service attacks using client puzzles. Proceedings of the 2011 IEEE International Conference on Web Services (ICWS'11), July 4-9, 2011, IEEE, Washington DC., USA., ISBN:978-1-4577-0842-8, pp: 25-32.

Wang, B., Y. Zheng, W. Lou and Y.T. Hou, 2015. DDoS attack protection in the era of cloud computing and software-defined networking. Comput. Netw., 81: 308-319.