ISSN : 2394-2975 (Online)
ISSN : 2394-6814 (Print)
Vol. 2, Issue 1 (Jan. - Mar. 2015)
International Journal of Advanced Research in Education Technology (IJARET)

Datalytix for Intrusion Detection of Botnet using Packet Inspection in Peer to Peer Network

[I]Rai Prajwal Ganesh, [II]Manjunatha A. S.
[1]M.Tech Student, [2]Senior Assistant Professor
[I,II]Dept.of CSE, Mangalore Institute of Technology and Engineering, Mangalore, Karnataka, India

Abstract
In Cybercrime, an increasing number of attacks are reported coming from Botnets. Botnet is one of the name which hover serious security threats on the current Internet infrastructure. Peer-to-Peer Botnets is a group of negotiated computers which are remotely controlled by its master under a decentralized Command and Control architecture. Botnets such as Spybot, Sinit and Zeus are popular, however newly constructed bot differ significantly employing improved techniques makes challenging task in botnet detection. This paper proposes and implements a framework for detecting Peer-to-Peer Botnet in live network and integrating with firewall to isolate compromised machine. Open source tools like Hadoop, Hive and Mahout are used for big data analysis for intrusion detection. Machine Learning library from Mahout is used for parallel processing of network traces to build Random Forest for Botnet detection in live network and to prevent further association from Botnet using Firewall.

Keywords
Big Data, Intrusion Detection System, Peer-to-Peer, Hadoop, Hive, Mahout, Firewall, Botnet Detection, Network Security.

I. Introduction

Peer-to-peer networks are networks in which all peers cooperate with each other to perform a function in a decentralized fashion. All peers are both client and server of services and can communicate with each other directly without intermediary agents. When compared with a centralized system, a Peer-to-Peer (P2P) system provides an easy way to aggregate large amounts of resource residing on the edge of Internet or in ad-hoc networks with a low cost of system maintenance and ease of performing malicious activity is a threat for security of Peer-to-Peer systems. Since there is no central server in most Peer-to-Peer systems, peers organize themselves to store and manage trust information about each other fig.1.

The construction of Peer-to-Peer networks is on the top of Network layer from OSI model, usually with a decentralized protocol allowing 'peers' to share resources. Perhaps, the huge success of Peer-to-Peer applications is mainly for to the ease of resource sharing provided by them in the form of video, music, files (Torrent), sharing of computing resources. Apart from these applications, Peer-to-Peer paradigm has also been widely deployed for LiveStation and Voice-over-IP (Skype) based services.

As Peer-to-Peer networks are inherently modeled without any centralized server, they lack a single point-of-failure. This flexibility given by Peer-to-Peer networks has attracted the attention of hackers in the form of bot-masters. A 'bot' is a computer program which allows the operative to remotely get control over the compromised system where it is installed. A network of such infected end-hosts under the remote command of a bot-master is called a 'Botnet'. With the capability of controlling massive number (in thousands) of bots in the network of computers, bot-masters gets the power of deploying attacks like click-fraud scam, mail spamming, Bitcoin mining, Distributed Denial of Service (DDoS) attacks etc. on a vast scale, and in turn generate millions of dollars per year in revenue for the bot-master fig.2.

In cyberspace, there exist three style of Botnet

- Centralized
- Peer-to-Peer
- Hybrid

Traditional Botnets were known to use Internet Relay Chat (IRC), which runs 'Command & Control (C&C) operations in a centralized architecture. Bot-masters have improved significantly ways of attacks by utilizing the resilient feature offered by Peer-to-Peer networks to build Botnets wherein bots communicate, pass on commands and update other bots in a Peer-to-Peer fashion. Just as a Peer-to-Peer network is resilient to break-down if a few peers leave the network, Peer-to-Peer Botnets have proven to be highly resilient even if a certain number of bots are identified and taken-down.

A Peer-to-Peer bot's life cycle consists of the following stages:

- **Infection stage**: stage where the bot spreads (this might happen through a malicious software being installed by the end-user, drive-by downloads, infected USB, etc.).
- **Rally stage**: during which the bot contacts series of hosts programmed to request configuration file, binary updates and neighbor peer lists, in order to join the Peer-to-Peer network.
- **Waiting stage**: is stage where the bot will be passive and waits for the bot-master's command.
- **Execution stage**: in this stage bot actually carries out a command, such as a Distributed Denial of Service (DDoS) attack, generates spam emails, etc.

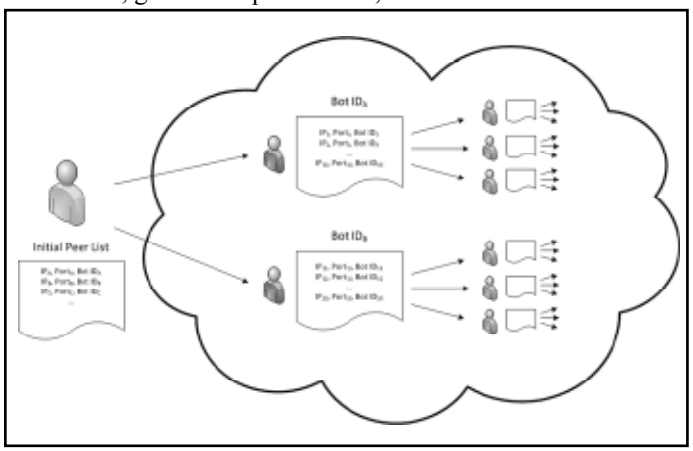

Fig.1: The process of Botnet spreading in network

ISSN : 2394-2975 (Online)
ISSN : 2394-6814 (Print)

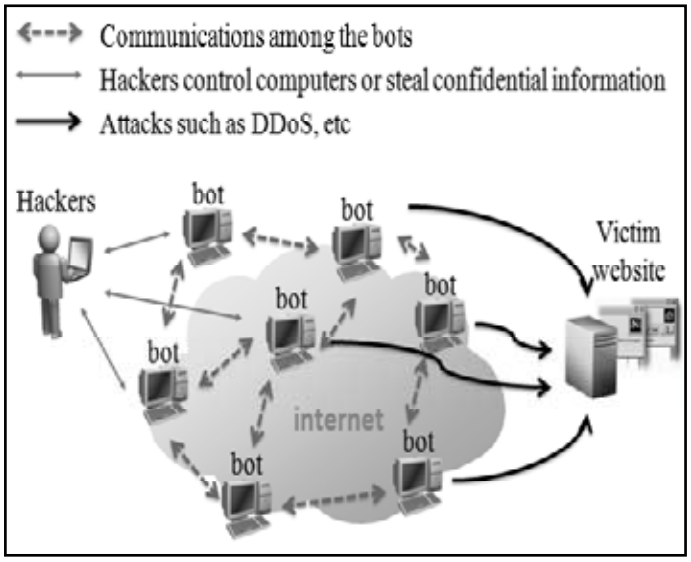

Fig. 2: Peer-to-Peer Botnet way of attack

The biggest tests that security faces today on global scale are Botnet attacks. Retailer's credit card user's account was stolen with help of Spyware Botnet. Numerous WordPress user accounts were lost because of enormous DDoS attack. Online security communities were suspicious it to be a Peer-to-Peer based Botnet. Botnet depends on compromised systems to send email spam attack using different messages.

To evade detection by Intrusion Detection Systems (IDSs) and Firewalls, communication patterns (with the bot-master or other bots) are quite stealthy by Botnet. IDSs and Firewalls which are our shield for protecting system from malicious attack rely on anomalous communication patterns to detect malicious behavior of a host, however IDS and Firewall are not very effective in detecting such Botnets which are very furtive and lie low, because they generate little traffic. Only way to detect and mitigate Bonet attacks is to trace network for packets. In this present generation packet flow as increased exponentially, attack from Bonet as increased too. However, computer systems lack the hardware and memory. This has led to an increase of interest in distributed algorithms, taking advantage of the multi-core architectures and distributed computing.

Network traces and capture packet are the valuable resources for detecting and blocking such attacks. Detecting bots individually is challenging, detecting them as cumulative holds the key. As attacks are increasing every day, size of network traces and capture packets are increasing exponentially. Standalone system lacks the hardware configuration for handing big data. However if single system is used for detecting Botnet by taking big dataset as input, dataset will itself acquire the whole heap size from the main memory, there will be no space for running JVM and running single process for the same will be slow to get final result. This led to advancement in distributed programming models, with the advantage of multi-core architecture, distributed processing and storage such as MapReduce, HDFS respectively.

Thus this paper proposes intrusion detection system for scalable and disturbed processing for big data and integrating with Firewall for blocking the Botnet from system. This framework uses Hadoop, which is open source software providing parallel and distributed computing using the power from the nodes involved in cluster. This framework utilizes Apache Hive for processing the packets. Apache Mahout has machine learning algorithm libraries to build classification prediction method.

The paper is organized as follows: Section 2 describes related work in field of machine learning algorithms for detecting Peer-to-Peer Botnet and Hadoop tool effectiveness for intrusion detection system. Section 3 concentrates on framework and briefly describes methodology to achieve real time Botnet detection. Section 4 describes application implementation for detection using the framework. Section 5 result and conclusion with future work.

II. Related Work

In recent years, machine learning approach is proposed for mitigating network security threats. Researchers find semantic-based method more efficient to detecting attacks compared to signature-based method [8]. In [5] researcher have proposed Hadoop based method to detect HTTP flood using Mapreduce on distributed computing for huge amount of traffic in reasonable time.

In field of security threat detection using machine learning approach there has been significant research. Author [6] proposes to design anomaly intrusion detection using machine learning and pattern recognition method.

In the paper [9] based on certain features which are related to traffic, the author has differentiate the network flow records. They also categorized them as Peer-to-Peer malicious and benign and also presented how plotter keeps changing their behavior to evade detection. This technique was also observed in Nugache behavior. Authors in [10] describe the challenges and features while handling Nugache Peer-to-Peer Botnet. Researchers [11] conclude detecting Nugache traffic using static intrusion detection is impossible.

Authors in [12] explain trust model can mitigate attacks and detect malicious peers. Reseachers [13] proposed technique to detect Zeus Botnet through network analysis. Authors studied Storm in [11] concludes that configuration file used by bot can be used to detect bot by IDS. However it is difficult to differentiate between authentic Peer-to-Peer communication and Storm bot's communication since the behavior of Storm can be disguised to look like authentic Peer-to-Peer communication. Authors [14] to address this issue, develop way to mitigate the Storm worm and introduce an active measurement technique to enumerate the number of infected hosts by reverse engineering of bot's binary executable, in order to identify the function which generates the key that is used for searching other infected machine and bots.

Researchers [15] retrieve the hashes of malware and use it for locating a zombie node's activities in Peer-to-Peer network. They proposed that if peer searches for hash malware, it must be zombie. Authors [16] formulate technique to localize Botnet members based on structured layout topologies used to communicate command and control.

Researchers [17] present a framework named BotMiner, which detect both centralized IRC and P2P Botnets using an anomaly-based detection system. The assumption in this regard is that bots are coordinated malware that exhibits similar communication patterns and behaviors. BotMiner targets a group of compromised systems belonging to monitored network, whereas it fails to detect a simple system which might be part of Botnet which is not the monitored network's zone.

In previous works, researchers have focused on detecting a specific Botnet activity and their methods were not successful in detecting bots, here traffic characteristics were not used in the training set. It can be seen that using machine learning based approach is

better in detecting malicious traffic when compared to traditional signature based approach as the bot masters redesign the bots from time to time. The functionality and behavior of the Botnet varies importantly with release of each version of the bot. Signature based detectors rely heavily upon the existing Botnet signatures to detect any activity. Mainly when handling zero-day attacks, signature based approach fails completely as there is no account of previous activity for that bot. Therefore, a machine learning approach is preferred to detect evidently suspicious activity based on the anomalous behavior of the network. Also in previous work, there has not been much research on deploying the detection module in a real-time scenario to monitor and mitigate Botnet activity in a network. In this paper, the handling of large-scale network traffic in a very short time is addressed and a solution is proposed to deal with Botnet detection at quasi-real time in heavy bandwidths of data traffic.

III. Methodology

A. Traffic Capture

For experiment, Network trace files (pcaps) are acquired from the sample of Peer-to-Peer bots obtained from [18, 19] which were deployed in a control environment test bed (as described section 4). Tshark[7] is a network protocol analyzer which uses pcap library to read and capture packets from live network. Dumpcap[4] is network traffic tool that uses Libpcap libraries. Dumpcap is used to sniff the packets from the network interfaces and while Tshark is used to extract and delimit the required fields out of the packets which are required for generating feature set for Machine Learning Module input. Dumpcap is used for capturing the packet from live network as it uses kernel level libpcap libraries. Dumpcap has better performance compare to Tshark, in case of buffer size to handle captured traffic and saving onto consecutive pcaps file.

Traffic Capture Module uses ring buffer option of Dumpcap to save the traffic packets monitored from the live network onto consecutive pcaps files of specified size. Tshark in parallel will extract the fields required and make a delimited file with specified separator for the feature set generation from the captured pcap files. These two processes are automated using Perl scripts for sniffing packets and extracting fields from them. The delimited files from created by Tshark instances are loaded into HDFS upon completion for feature extraction.

B. Feature Extraction

Once HDFS as the delimited file, next step is to extract feature sets by using Apache Hive [2].Hive an open-source data warehouse system on top of Hadoop[1] for querying and analyzing large datasets similar to relational database management system (RDBMS). Hadoop is a framework using Mapredue for handling large datasets in a distributed computing environment.

Apache Hive is used in this framework to add the flexibility of changing the feature extraction at run time with respect to Tshark extracted fields. As mentioned above, Perl script is used to sniff and extract fields and create Hive table accordingly to the fields decided by user to extract from packet. Importance of using have here is to avoid low level Mapreduce programming which requires developers to write customs programs which are hard to reuse by changing code dynamically.

Apache Hive facilitates easy extract/transform/load (ETL). Hive uses a specific query language called as HiveQL (Hive Query Language) is similar to SQL declarative language and therefore can be easily learnt by people acquainted with SQL [2]. HiveQL are compiled into Mapreduce jobs in runtime. The map phase transforms the input into key-value pair based on the query and passed to reducer phase. Then the reducer aggregates value by groups of key and stores the output.

Basically, the user connects to the user interface and executes a HiveQL command, which is sent to the driver. The driver then creates a session and sends the query to the compiler which extracts metadata from the MetaStore and generates an execution plan. This logical plan is then optimized by the query optimization component of Hive and translated into an executable query plan consisting of multiple map and reduce phases. The plan is then executed by the MapReduce execution engine consisting of one jobtracker and possibly several task trackers per map and reduce phase.

Table below lists the features extracted from the table using "group by" clause in HiveQL. This list is the network flow statistics of packets transferred and it's organized as group of source ip, source port, destination ip, destination port and protocol.

Table 1: Feature statistics extracted from the software

Feature	Description of the feature
Srcip	Source ip address (string)
Srcport	Source port number
Dstip	Destination ip address (string)
Dstport	Destination port number
Proto	Protocol (tcp=6 or udp=17)
total_fpacket	Total packets in forward direction
total_bpackets	Total packets in backward direction
total_bvolume	Total bytes in backward direction
min_fpktl	Size of smallest packet sent in forward direction (in bytes)
mean_fpktl	Mean size of packets sent in forward direction (in bytes)
max_fpktl	Size of largest packet sent in forward direction (in bytes)
std_fpktl	Standard deviation from mean of packets sent in forward direction (in bytes)
min_bpktl	Size of smallest packet sent in backward direction (in bytes)
mean_bpktl	Mean size of packets sent in backward direction (in bytes)
max_bpktl	Size of largest packet sent in backward direction (in bytes)
std_bpktl	Standard deviation from mean of packets sent in backward direction (in bytes)
min_fiat	Minimum amount of time between two packets sent in forward direction (in microseconds)
mean_fiat	Mean amount of time between two packets sent in forward direction (in microseconds)
max_fiat	Maximum amount of time between two packets sent in forward direction (in microseconds)
std_fpktl	Standard deviation from mean amount of time between two packets sent in forward direction (in microseconds)
min_biat	Minimum amount of time between two packets sent in backward direction (in microseconds)
mean_biat	Mean amount of time between two packets sent in backward direction (in microseconds)

max_biat	Maximum amount of time between two packets sent in backward direction (in microseconds)
std_biat	Standard deviation from mean amount of time between two packets sent in backward direction (in microseconds)
Duration	Duration of flow (in microseconds)
fpsh_cnt	Number of times PSH flag was set in packets travelling in forward direction (0 for UDP)
bpsh_cnt	Number of times PSH flag was set in packets travelling in backward direction (0 for UDP)
furg_cnt	Number of times URG flag was set in packets travelling in forward direction (0 for UDP)
burg_cnt	Number of times URG flag was set in packets travelling in backward direction (0 for UDP)
total_bhlen	Total bytes used for headers in backward direction
total_fhlen	Total bytes used for headers in forward direction

C. Machine Learning

Apache Mahout [3] is package of implementations of the most popular machine learning algorithms, with the implementations designed specifically to use Hadoop to enable scalability in processing of huge data sets. As Mahout Machine Learning library algorithms are run as Mapreduce jobs, it uses power of distributed computing from the cluster to obtain efficient result.

Classification is a simplified form of decision making that gives discrete answers to an individual question. Machine-based classification is an automation of this decision making process that learns from examples of correct decision making and emulates those decisions automatically, it's a core concept in predictive analytics and is an instance of supervised learning i.e. learning where a training set of correctly identified observations is available.

The Random Forest algorithm's is used from Mahout's library for training model and classification purpose. Random forests trains many number of classification trees. It is an ensemble learning method for classification that constructs a number of decision trees at training time and give result as class that is the mode of the classes predicted by individual trees. Random Forest unlike single decision trees does not suffer from high variance, because it finds average collection of decision trees.

Random Forest algorithm has lot of benefits in classification like combining result from learning model increases the accuracy of the classification, runs effective on large datasets and is time efficient as well as can handle a large amount of explanatory variables, that is why Random Forest becomes perfect choice for Botnet detection.

First a file descriptor is generated for the dataset got from the previous module (feature extraction) before building Random Forest model into three attribute (numeric, catergorical and label).-Dmapred.max.split.size argument is used to control maximum size of each partition (1/10 of size of dataset) by Hadoop.[20] Thus 10 partitions will be used. Then 100 trees are built, each tree using 5 random selected attributes per node and predicted outputs of tree is stored onto HDFS from which malicious node can be identified.

D. Firewall Rule

Once the malicious node is identified by previous module, its ip address is obtained from the result which is stored in HDFS. IPtable firewall is used to manage packets filtering for compromised system ip address. IPtable tool is used to manage the Linux firewall rules. On high-level iptables might contain multiple tables and those tables might contain multiple chains. In turn chains might contain multiple rules. Chains can be built-in or user defined. Rules are defined for the packets. [21]

Firewall IPtable shell script is used to add the rules. These rules are set for the incoming and outgoing packet from particular system. Rules will contain target value as accept or drop for the mentioned protocol, source and destination.

IV. Experimental Setup

In order to contain and collect malicious source for this research work, test bed consists of standalone network of Linux systems. The systems were connected to an Access switch in order to form the isolated network. On top of each of these physical machines, windows operating system is run as guest OS on virtual machine. This virtual machine has Peer-to-Peer Botnet samples deployed on. The samples Botnet are Waledac, Zeus, Kelihos, Conficker [].

The network was monitored for malware sample activity and captured by Dumpcap for 48 hours each. After collecting packet traces of network activity right from the infection stage to the attack stage of each malware, the pcap files were stored in HDFS for further analysis. Dumpcap uses ring buffer to create successive pcap files, an optimal size of 50 MB until the condition is satisfied, in this case is the time mention.

To train classification module in efficient way, captured file from known Bot attacks are used as dataset. These datasets also contain benign traffic which is needed for classifier such as ftp transfer, video streaming, p2p application and telnet session. These entire datasets are fused and used from machine learning algorithm for model generating, from which 90% is taken as training set and 10% is taken as testing set. For training the framework dataset from ( ) is replayed onto the network using TCP Replay. The sequence of steps used in this real-time Peer-to-Peer Botnet detection and isolated from network is shown in fig 3.

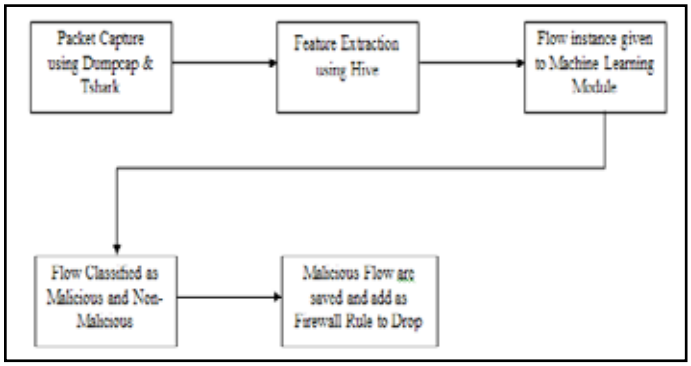

Fig. 3: Overview of Real-time Deployment of the Detection Module

V. Conclusion and Future Work

It is introductory state of the scheme presented in this paper, a lot of ground-work still needs to be done. This paper puts forward methods and flow to find abnormalities in network traffic data using data big analytics and distributed computing for faster result. The ensemble trees classification and prediction the rule set in Firewall is updated. In this paper we presented the methods to

detect Botnet DDos and communication that is recognized by machine learning module using Random Forest algorithm. This method can help cover the gap in traditional network systems, to achieve a better performance and detection of Peer-to-Peer Botnet. It is expected that future work on this IDS can be done by analysing more advanced algorithm and get more efficient results in network traffic using patterns from Hadoop.

VI. Acknowledgment

I am very thankful to my guide Mr Manjunath A. S. Senior Assistant Professor, Department of Computer Science and Engineering, Mite for his cordial support, valuable information and guidance, to prepare this paper and also thankful to Prof. Dr. Nagesh H R, Head of the Department, Computer Science and Engineering, for his valuable and constructive suggestions during the planning and development of this work.

References

[1] *Apache, Hadoop, http://hadoop.apache.org/*
[2] *Apache, Hive, https://hive.apache.org/*
[3] *Apache, Mahout, http://mahout.apache.org/*
[4] *Dumpcap, http://www.wireshark.org/docs/ man-pages/ dumpcap.html*
[5] *Yeonhee Lee, Youngseok Lee," Detecting DDoS attacks with Hadoop", Article No. 7 ACM New York, NY, USA, 2011 table of contents ISBN: 978-1-4503-1042-0.*
[6] *Phyu Thi Htun, Kyaw Thet Khaing, "Anomaly Intrusion Detection System using Random Forests and k-Nearest Neighbor" Journal IJPTT, Volume-3,issue-1, IJPTT-V3I1P413*
[7] *Tshark, https://www.wireshark.org/docs/ man-pages / tshark. html*
[8] *A.Razzaq, K. Latif, H.F Ahmad, A. Hur, Z. Anwar, P.C. Bloodworths, "Semantic security against web application attracks. Information Science 254, 2014*
[9] *T.F. Yen, M.K Reiter, " Are your hosts trading or plotting? Telling P2P file-sharing and bots apart" IEEE, 2010*
[10] *D. Dittrich, S. Dietrich, "P2P as Botnet command and control: a deeper insight", 3rd International Conference on Malicious and Unwanted Software, MALWARE 2008, IEEE, October 2008*
[11] *S. Stover, D. Dittrich, J. Hernandez, S. Dietrich, "Analysis of the Storm and Nugache trojans: P2P is here ", Usenix; Login 32, 2007*
[12] *D. Arndt, Netmate, http://dan.arndt.ca/nims/calculating-flow-statistics-using-netmate*
[13] *Benvenuti, Understanding Linux Network Internals, 2009.*
[14] *Holz, T.M Steiner, F. Dahl, E. Biersack, F. Freiling, "Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm", Usenix workshop,2008*
[15] *J.B. Grizzard, V. Sharma, C. Nunnery, B.B. Kang, D. Dagon, "Peer-to-peer botnets: overview and case study", Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets, 2007*
[16] *S. Nagaraja, P. Mittal, C.Y. Hong, M. Caesar, N. Borisov, "BotGrep: finding P2P bots with structured graph analysis", USENIX Security Symposium, 2010*
[17] *G. Gu, R. Perdisci, J. Zhang, W. Lee, "BotMiner: clustering analysis of network traffic for protocol-and structure-independent botnet detection", USENIX, Security Symposium, 2008*
[18] *ContagioDump Blogspot, http://contagiodump.blogspot.in*
[19] *Open Malware, http://openmalware.org/*
[20] *Mahout Partial Implementation https://mahout.apache.org /users/classification/partial-implementation.html*
[21] *Firewall Rules, http://www.thegeekstuff.com/2011/02/ iptables-add-rule/*
[22] *Kamaldeep Singh, Sharath Chandra Guntuku, Abhishek Thakur, Chittaranjan Hota, "Big Data Analysis framework for peer-to-peer detection using random forest", information science, 2014*

Authors Profile

Rai Prajwal Ganesh completed the Bachelor's Degree in Computer Science & Engineering from Visvesvaraya technological University (VTU). Currently pursuing M.Tech degree in Computer Science & Engineering at Mangalore Institute of Technology, Mangalore.

Mr. Manjunath A. S. received Master Degree in computer science and Engineering. He is currently working as Senior Assistant professor in the Department of Computer Science and Engineering, Mangalore Institute of Technology and Engineering, Mangalore.