International Journal of Electronics and Applied Research (IJEAR) **vol. 4, issue 1, July 2017**
Online (*http://eses.net.in/ESES Journal)* ISSN 2395-0064

Cyber pirating and Detection of malicious activities p2p botnets using Markov cluster algorithm

Dhruba Jyoti Borah and Abhijit Sarma
Department of Computer Science. Gauhati University, Guwahati 781014, India

Abstract

Botnet is a network of compromised computers connected through Internet and performs malicious activities such as, stealing passwords, breaking down the security system of some confidential websites, performs DDoS attacks, web account abuse attacks, click frauds, information leakage, etc. Botnet uses random ports, encrypts its content to evade detection. P2P botnet uses same communication protocols as the normal P2P applications use and thus it is difficult to extract its traffic pattern from regular traffic. In this paper, we use graph mining approach to detect botnet by analysing the communication structure of a network. We use netflow data to create a host dependency graph and then use Markov cluster algorithm to detect clusters that contain bot nodes. Experimental results indicate that our technique can localize a majority of bot nodes with very low false positive rate.

Key words: Botnets; Markov Cluster Algorithm; Network security; Graph mining

1. Introduction

Botnet is a network of infected hosts controlled by hackers, known as botmaster. The botmaster may use his network for performing various malicious activities such as, stealing passwords, breaking down the security system of some confidential websites, performs DDoS attacks, web account abuse attacks, click frauds and information leakage, etc. [Yao Zhao et al., 2009; BotTrack: tracking botnets using Net Flow and PageRank, Jérôme François et al., 2011; Wernhuar Tarng et al., 2011]. The botmaster creates some pieces of software, known as bots. These bots are installed in the vulnerable computers using similar techniques as used by virus, Trojan horse, etc. After installation, these bots collect confidential

https://doi.org/10.33665/IJEAR.2017.v04i01.001

information from the infected computers and send them to the command and control (C&C) server of the botnet as required by the botmaster. The botmaster may give commands through the command & control (C&C) server to these bots to attack other personal computers, web servers, etc. Since these infected computers can be used by the botmaster as his army, they are also known as zombie computers.

There are two types of command & control (C&C) server; centralized C&C and decentralized C&C. In centralized C&C server, there is a central C&C server, through which the botmaster gives command to the zombie computers. In IRC and HTTP based botnets, the botmaster uses central C&C server [Wernhuar Tarng et al., 2011]. In IRC botnet, the botmaster uses IRC servers to communicate with his army. EggDrop, Gtbot Variants are some examples of IRC based botnets [Julian B. Grizzard et. al., 2007]. The botmaster can setup his own IRC server or use public IRC server to give commands to the zombie computers. The zombie computers are automatically connected with IRC server and wait for the next instruction to execute. EggDrop was the first IRC based botnet appeared in 1993[Julian B. Grizzard et al., 2007]. It was developed for simply improving the automation on the Internet. But subsequently, non-ethical activities were performed through EggDrop. So, it was recognized as a malicious botnet. HTTP botnet is similar to IRC based botnet. But in HTTP botnet, the botmaster gives instructions to his armies by using some malicious HTTP servers.

Although both IRC and HTTP based botnets are convenient to perform malicious activities, these are not popular botnets in today's internet world. They use client-server architecture and thus have a single point of failure. This means that, both these botnets have central command and control (C&C) server. If the server is shut down, the entire network collapses.

Therefore, now a days, decentralized C&C servers are more used by the botmasters. Peer-to-peer (p2p) botnet uses decentralized C&C server. Here, all the zombie computers act as peers to others. The botmaster uses any one of them as C&C server. He can switch the C&C server from one zombie computer to other. All the peers listen to their peers to get any instruction from the C&C server. Figure 1 (a) Shows the structure of centralized botnet and Figure 1(b) shows the structure of decentralized botnets.

Some notorious botnets which are dismantled now are Rustock, Donbot, Srizbi, Mariposa, Waledac, BredoLab, Kelihos, etc. [https://en.wikipedia.org/wiki/Botnet]. Festi, Vulcanbot, Mirai (malware), Zeus GameOver are some examples of botnets which are still active and cause loss of hundreds of million dollars. Recently, researchers from antivirus provider Eset, discovered a backdoor Trojan of a p2p malware called Turla [https://arstechnica.com/security/2017/06/russian-hackers-turn-to-britney-spears-

for-help-concealing-espionage-malware/]. Turla is a Russian speaking hacking group that targets governments. The Trojan of the Turla used comments posted in the photos of official Instagram account of Britney Spears to locate the C&C server of the network.

In this work, we focus on detection of p2p botnet by analyzing the communication traffic of a network. We use netflow data, since it is more reliable to efficiently evaluate large volume of data. Our main goal is to design a mechanism to detect bot nodes of various botnets by using Markov cluster algorithm with very low false positive rate.

2. Related work

Based on the different types of algorithms used by the researchers to detect botnets, the author of [Atef Ahmed Obeidat, 2016] differentiates them into three types, which are (a) neural network-based algorithms, (b) Heuristic threshold-based algorithm, (c) Mining-based algorithm. The neural based algorithms build supervised machine learning models by using neural networks to detect p2p botnets. Multiple machine-learning algorithms such as, Bayesian networks, decision trees are used to validate the approaches. One example of neural network-based algorithm is used by [Wernhuar Tarng et al., 2011], where a J48 decision-tree model is developed. This algorithm can detect not only known bots but also unknown bot nodes. After detection of unknown botnet, its traffic sample can also be used to train the J48 decision-tree model. But this mechanism was experimented in a very small environment. Therefore, this mechanism may not perform well in a very large network. The heuristic threshold-based algorithms use some threshold values to differentiate various characteristic between bot nodes and benign nodes. For example, this type of algorithm is used by [Connor Dillon, 2014]. The author of [Connor Dillon, 2014] tries to detect Zeus p2p botnet. Here he calculates the packet ratio, which is the sum of up packets divided by sum of down packets. The p2p applications which have the packet ratio less than threshold value 1.4 are considered as botnet. Mining based algorithm uses data mining technique to detect botnet. For example, [Yao Zhao et al., 2009; BotTrack: tracking botnets using NetFlow and PageRank, Jérôme François et al., 2011; Shishir Nagaraja et al., 2010; BotCloud: Detecting botnets using MapReduce, Jérôme François et al., 2011] uses graph mining technique to detect botnet. [Shishir Nagaraja et al., 2010] uses random walk on the host dependency graph. On the other hand, [BotTrack: tracking botnets using NetFlow and PageRank, Jérôme François et al., 2011] uses pagerank algorithm [Anand Singh Kunwar,

1999] on the communication graph and then uses DBScan clustering algorithm to detect botnet. On the other hand, if a group of hosts share similar communication patterns and perform similar malicious activities, BotMiner [Guofei Gu et al., 2008] identifies them as bots belonging to the same botnet. But, the malicious activity can be very stealthy and non-observable. Therefore, BotMiner is ineffective now a days. The authors of [Ping Wang, Lei Wu, Baber Aslam, 2009] give two defence mechanisms: index poisoning and Sybil attack. Index based mechanism is used as a defence mechanism against index based p2p network. It does not give fruitful result in other types of p2p botnets. In Sybil attack, multiple identities are forged to subvert the reputation system in a p2p network. It is possible to do so in a p2p network. Because peers can join in a p2p network without proving its identity. But it is a challenging task to add multiple identities into a p2p botnet.

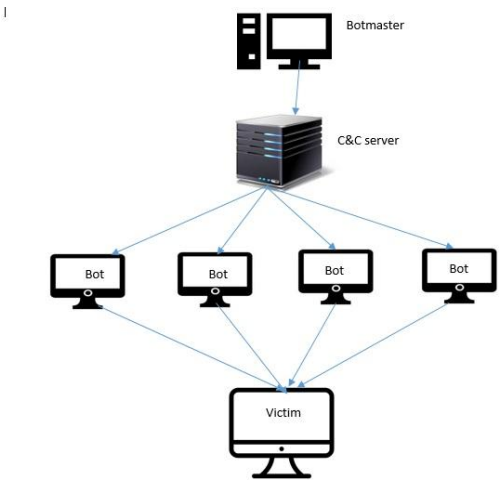

Figure 1(a): Structure of centralized botnet

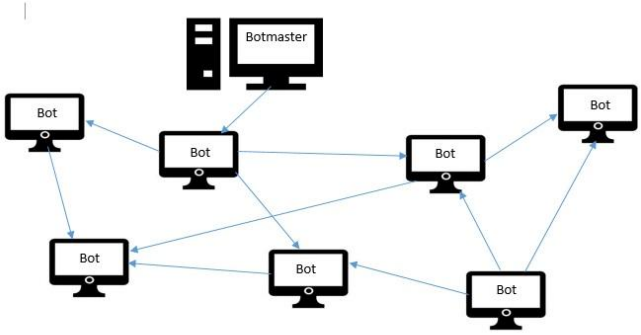

Figure 1(b): Structure of decentralized (p2p) botnet

3. System Design

Figure 2 shows our approach to detect botnet. At first netflow data are exported from the routers to a collector. Then a host communication graph is built by analyzing the data. This graph represents which nodes are communicating among themselves. Then this graph is used as an input to the Markov cluster algorithm [Stijn van Dongen, 2000]. The Markov cluster algorithm gives a set of clusters as output

containing all the nodes (including bot nodes) of the graph. Then a prior knowledge is used to determine the clusters containing bot nodes.

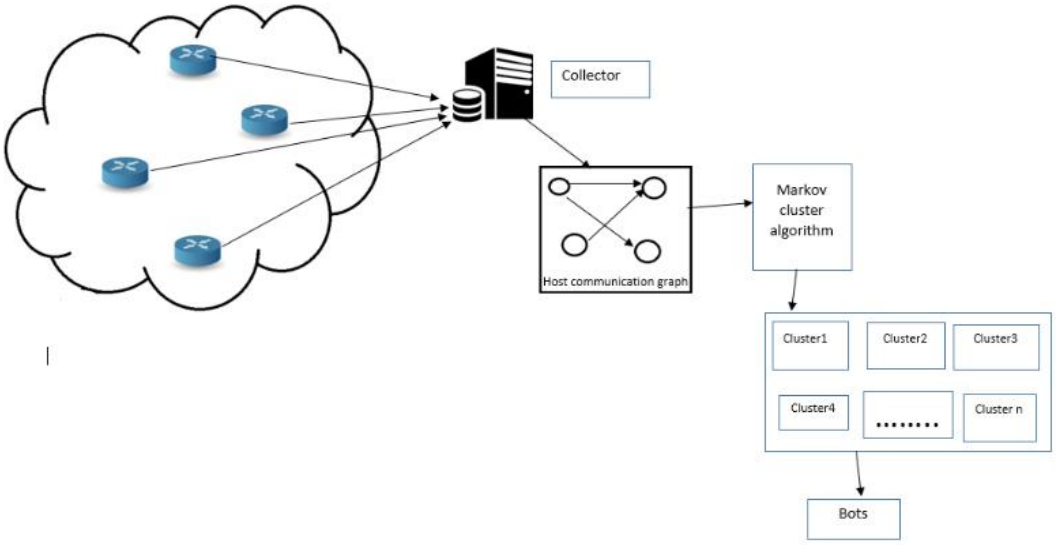

Figure 2: Our approach to detect botnet

3.1. NetFlow data

NetFlow [https://tools.ietf.org/html/rfc3954] is a standard tool developed for collecting and monitoring today's large-scale network traffic. By analyzing flow data, a picture of network traffic flow and volume can be built. A record of netflow data contains information, such as packets and bytes counts, timestamps, type of service, IP addresses, ports, output interfaces, etc. of a series of IP packets sharing same source IP address and port and destination IP address and port. Although it was developed by Cisco, now it is supported by almost all modern commercial-grade network elements (routers and switches).

Analyzing network traffic using netflow data is more manageable than using each packet of the network. This is because, netflow data does not contain the payload of any IP packet. Therefore, it needs a fraction of memory for full packet capture. Hence it reduces time to analyse network traffic.

3.2. Host communication graph

Host communication graph is drawn on the netflow data collected from the routers. In this graph, each host is represented by vertex and each edge in the graph pointing from vertex A to vertex B implies that, there is at least one flow which originates from host A to host B. Hence, bots within the same botnet are linked with each other either directly or indirectly. Hence, host communication graph gives a clear picture of the communication of the nodes among themselves. Figure 3 shows a host communication graph of 10 nodes.

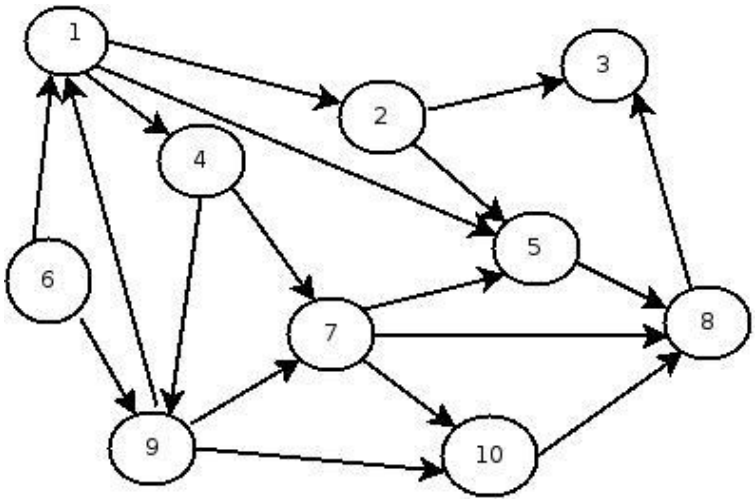

Figure 3: Sample host communication graph of 10 nodes.

3.3. Markov cluster algorithm

Markov cluster algorithm (MCL) [Stijn van Dongen, 2000] is a fast and scalable unsupervised cluster algorithm for graphs that clusters all the nodes of a graph into their desired clusters. In MCL, input is a transition probability matrix of the graph, where each column is sum up to 1. Figure 4(b) shows a transition probability matrix (also known as column stochastic matrix) for a sample input graph figure 4(a).

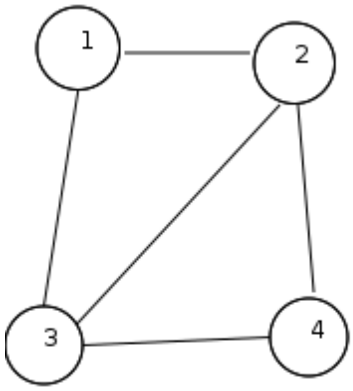

Figure 4(a): A sample graph.

	1	2	3	4
1		33	.33	
2	.5		.33	.5
3	.5	.33		.5
4		.33	.33	

Figure 4(b): Column stochastic matrix.

MCL simulates random walks within the input graph by altering two operations namely expansion and inflation to find the cluster structure in the graph. Expansion operator simply expands the matrix, using matrix multiplication. Inflation takes the hadamard power of a matrix (taking powers entry wise) and then perform a scaling procedure, such that the resulting matrix is also a stochastic matrix [https://micans.org/mcl/index.html?sec_description1]. The main idea of Markov cluster algorithm is to repeatedly apply the two operations until a stable state of the column stochastic matrix is reached. The stable state is the state from where the column stochastic matrix will not change its state, if we apply the two operations on it.

We use MCL algorithm for the following reasons:

(a) It is simple: This algorithm is uses only two simple algebraic operations, i.e. inflation and expansion. No high-level instructions are used in this algorithm.

(b) It is adaptable: There is no need to specify the number of clusters in advance. We can get different clustering on different scales of granularity by varying a single parameter.

(c) It is relatively fast: According to [https://micans.org/mcl/index.html/sec_description1], the complexity of an optimized MCL algorithm is $O(Nk^2)$, where N represents the number of nodes in the graph, and k represents the number of resources allocated per node.

3.4. Impact of known bots

After clustering the input graph, all the nodes of the graph are clustered. Since we do not know the exact position of the bot nodes after clustering, we consider that, the cluster which contains at least one bot node, all the nodes in that cluster are also bot nodes. Therefore, prior knowledge of some individual bots is needed to detect botnet. This advance knowledge can be given by some network traffic analysing tool, such as honeypots [Lance Spitzner, 2002]. This may lead to declare some benign nodes as bot nodes. But figure 5 shows that, our design declares very small number of benign nodes as bot nodes.

4. Experiments and results

To evaluate the performance of our mechanism, it has to be used in real network traffic. But, real network traffic with bot labels and non-bots is not available. Therefore, we developed a bot as per the specification of Zeus Gameover, obtained from CERT-Polska [https://www.cert.pl/wp-content/uploads/2015/12/2013-06-p2p-rap_en.pdf]. The bots were run on a testbed consisting of four blade servers, where 54 Linux virtual servers are installed. For non-bot traffic, we used CAIDA [http://www.caida.org/data/passive/passive_oc48_dataset.xml], anonymized OC48 Internet traces dataset (2002-2003). For better evaluation, we took five datasets of real-world network traces. Both the botnet traffic and the real network non-bot traffic are then mixed up, which were the actual botnet embedded traffic.

On those traffic data, a host communication graph, where each vertex represented a host of the network and each edge pointing towards vertex B from vertex A represented an outgoing link from host A to host B was created. This graph was sent as input to the MCL algorithm. MCL clustered all the nodes into their desired clusters. Then we saw that, in each of the five datasets, one cluster contained minimum 80% bot nodes.

We took help of Receiver Operating Characteristics (ROC) [Tom Fawcett, 2006] to evaluate our mechanism. ROC measures the true positive rate (TPR) against the false positive rate (FPR), where TPR is the number of correctly identified bots divided by total number of bots and FPR is the number of benign nodes identified as bots by our design divided by total number of benign nodes. The values of TPR and FPR of our experiment are as follows:

Table I

Results of our design

Dataset	Number of nodes in the dataset	TPR	FPR	FP	TP (%)	FP (%)	clusters
Dataset1	45981	0.91	0.00037	17	91.2%	0.0371%	9530
Dataset2	237017	.89	0.00169	4	88.89%	0.00169%	23853
Dataset3	296220	.78	0.00101	3	79%	0.00101%	25366
Dataset4	234783	.78	0.00128	3	79%	0.00128%	23804
Dataset5	265452	.93	0.0003	1	94%	0.0003%	17177

The graphical representation of all the clusters for the above five datasets are as follows:

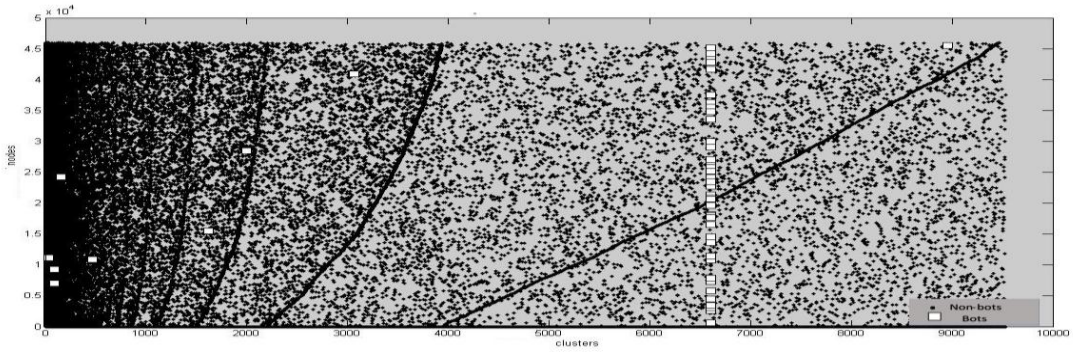

Figure 5(a): Results of dataset1 shows 91.2% bots are in the same cluster.

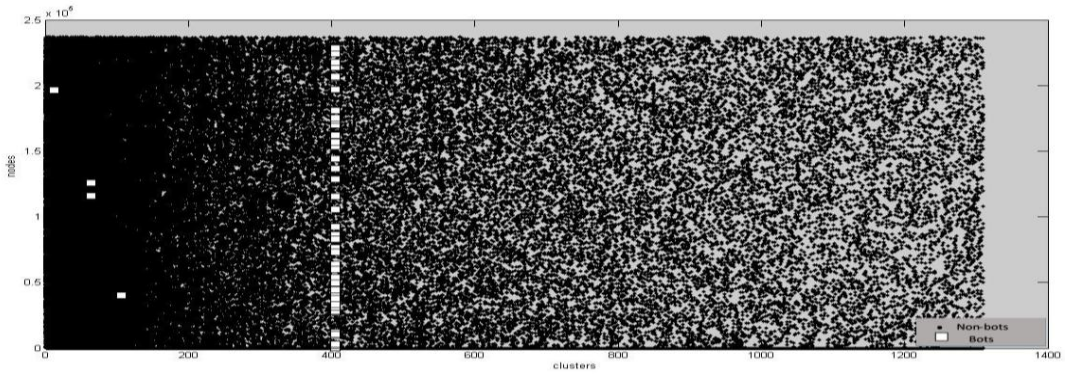

Figure 5(b): Results of dataset2 shows 88.89% bots are in the same cluster.

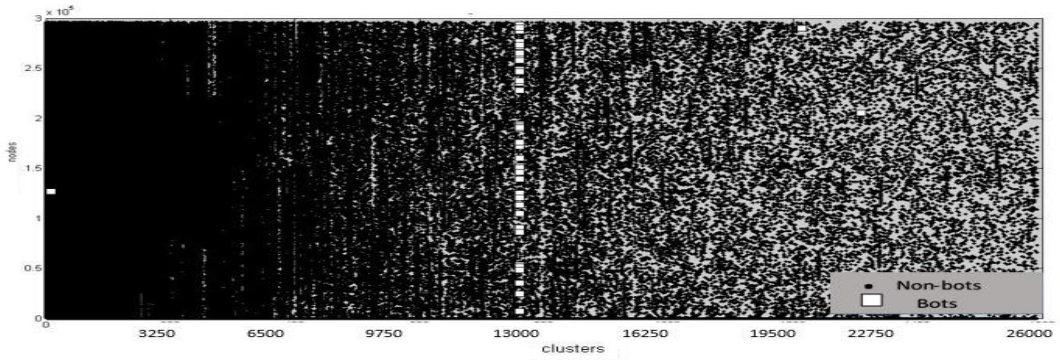

Figure 5(c): Results of dataset3 shows 79% bots are in the same cluster.

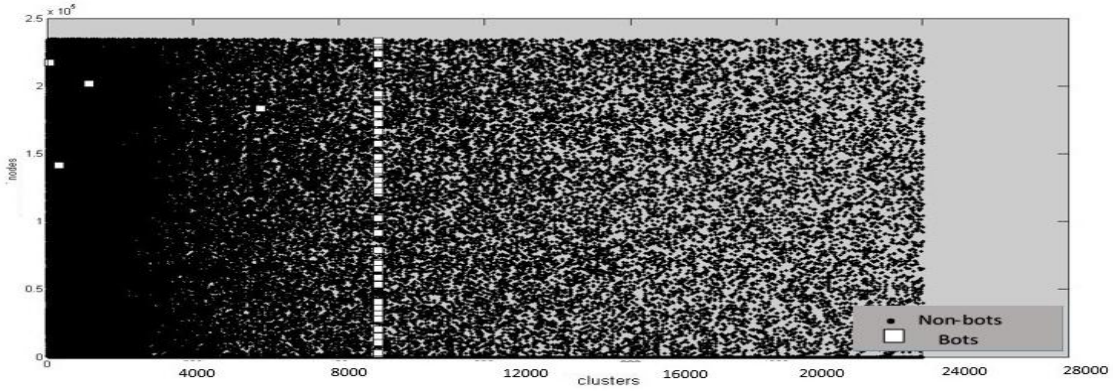

Figure 5(d): Results of dataset4 shows 79% bots are in the same cluster.

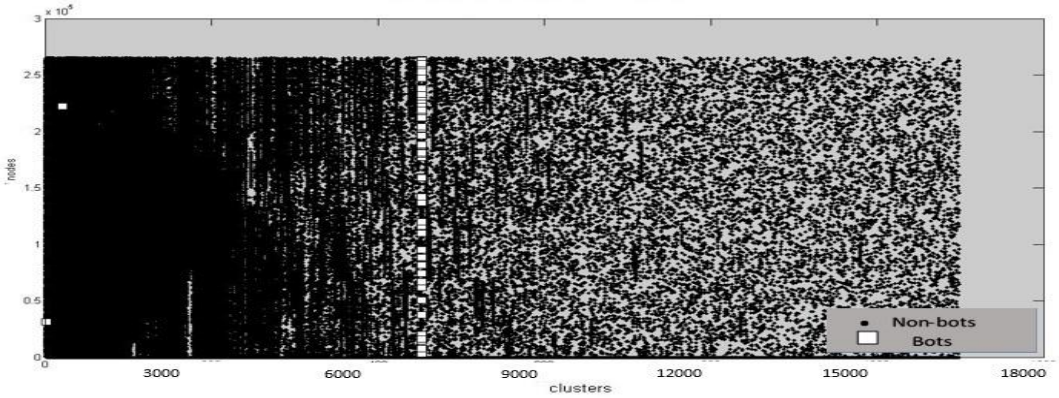

Figure 5(e): Results of dataset 5 shows 94% bots are in the same cluster.

It is seen that, the method can detect botnets up to 94%.

5. Conclusion

In this paper, we attempt to detect p2p botnets. Our detection mechanism takes netflow related information as input and build a host communication graph. This graph is then used as an input to the Markov cluster algorithm to cluster the nodes in the graph. By using some prior knowledge of bots, we have localized a majority of bot nodes with a very low false positive rate. Since botnet traffic is not available, we generated botnet traffic in our testbed and mixed them with some real-world traffic collected from CAIDA. We plan to detect botnet by modifying some characteristics of the Markov cluster algorithm. Because Markov cluster algorithm outputs many small clusters. Therefore, although all the peer nodes are tightly connected, some bot nodes are clustered into different clusters by Markov cluster algorithm. On the other hand, our future plan also includes detection of botnet without using any prior knowledge of botnet.

References

Botnet. https://en.wikipedia.org/wiki/Botnet, (2017, May 27). Retrieved May 28, 2017.

Claise, B., "Cisco Systems NetFlow Services Export Version 9," RFC 3954 (Informational), https://tools.ietf.org/html/rfc3954.

Dan Goodin - Jun 6, 2017 10:40 pm UTC. (2017, June 06). You'll never guess where Russian spies are hiding their control servers. from https://arstechnica.com/security/2017/06/ russian-hackers-turn-to-britney-spears-for-help-concealing-espionage-malware/, Retrieved June 14, 2017.

Dillon, C. Peer-to-Peer Botnet Detection Using NetFlow, *Master Thesis,University of Amsterdam*, 2014.

Dongen, S. V., Graph Clustering by Flow Simulation, (*PhD thesis) University of Utrecht, Netherlands)*, 2000.

Dongen, S. V. (n.d.). MCL- a cluster algorithm for graph: An introduction to MCL.,from https:// micans.org /mcl/index.html ?sec_description1, Retrieved June 05, 2017.

François, J., Wang, S., State, R., & Engel, T. BotTrack: tracking botnets using Net Flow and Page Rank, *Proceedings of IFIP/TC6 Networking*, doi: 10.1007/978-3-642-20757-0_1, 6640, 2011.

Fawcett, T. An introduction to ROC analysis, *Pattern Recognition Letters - Special issue: ROC analysis in pattern recognition,*Volume **27** Issue 8, 861-874, 2006.

Francois, J., Wang, S., Bronzi, W., State, R., & Engel, T. Botcloud: Detecting botnets using mapreduce. *In Information Forensics and Security (WIFS), IEEE International Workshop* , 1-6, 2011.

Grizzard, J. B., Sharma, V., Nunnery, C., Kang, B. B., & Dagon, D. Peer-to-Peer Botnets: Overview and Case Study. *Proceedings of USENIX HotBots'07,* 2007.

Gu, G., Perdisci, R., Zhang, J., & Lee, W. BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection. *In USENIX Security Symposium*, Vol. **5**, No. 2, 139-154, 2008.

Nagaraja, S., Mittal, P., Hong, C. Y., Caesar, M., & Borisov, N. BotGrep: Finding P2P Bots with Structured Graph Analysis. *In Proceedings of the 19th USENIX conference on Security*, 95-110, 2010.

Obeidat, A. A. Analysis the P2P botnet detection methods. *IPASJ International Journal of Computer Science (IIJCS)*, Volume 4, 1-11, 2016.

Page, L., Brin, S., Motwani, R., & Winograd, T. *The Page Rank citation ranking: Bringing order to the web*, Stanford InfoLab, 1999.

Spitzner, L., *Honeypots: Tracking Hackers*, (Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA), 2002.

Tarng, W., Den, L., Ou, K., and Chen, M. The analysis and identification of p2p botnet traffic flow, *International Journal of Communication Networks and Information Security (IJCNIS)*, Vol. **3**, No. 2, 138-148, 2011.

Technical report on ZeuS-P2P monitoring and analysis, https://www.cert.pl/wp-content/uploads/2015/12/2013-06-p2p-rap_en.pdf, 2015.

The Center for Applied Internet Data Analysis (CAIDA). (n.d.). from http:// www.caida.org/data/passive/passive_oc48_dataset.xml, Retrieved from May 05, 2017.

Wang, P., Wu, L., Aslam, B., & Zou, C. C. A systematic study on peer-to-peer botnets, *In Computer and Communications Networks, Proceedings of 18th Internatonal Conference*, doi:10.1109/ICCCN.2009.5235360, 2009.

Zhao, Y., Xie, Y., Yu, F., Ke, Q., Yu, Y., Chen, Y., Gillum, E. *BotGraph: Large Scale Spamming Botnet detection*, The 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI '09), 321-334, 2009.