Botnet Detection using NetFlow and Clustering

Botnet Detection using NetFlow and Clustering

Pedram Amini[1], Reza Azmi[2] and MuhammadAmin Araghizadeh[3]

[1] ICT Department, Malek-Ashtar University of Technology
Tehran, Iran
amini@mut.ac.ir

[2] Department of Technical and Engineering, Alzahra University
Tehran, Iran
azmi@alzahra.ac.ir

[3] Department of Electrical and Computer Engineering, University of Tehran
Tehran, Iran
araghizadeh@ut.ac.ir

Abstract

Among the various forms of malware, botnets are becoming the major threats on the Internet that use for many attacks, such as spam, distributed denial-of-service (DDoS), identity theft and phishing. NetFlow protocol is a standard for monitoring Internet traffic that developed by Cisco Systems. Therefore, it is very effective to identifying unusual programs generating illegal traffic, or additional load, and also identification of botnet. The main goal of this paper is to show a novel approach for botnet detection using data records of NetFlow protocol and clustering technique. Our approach for C&C bot detection is to examine flow characteristics such as IP, port, packet event times and bytes per packet for evidence of botnet activity. First we collect the flows and refined records based on basic filtering, white list and black list. The remaining records produce a cluster and the cluster refined based on patterns, policies, and another cluster that generated based on reported events, alerts and activities of network security sensors. We apply hierarchical clustering that allows us to build a dendrogram, i.e., a tree like graph that encodes the relationships among the bots. The merged cluster modifies based on rules and combined with other information about detected infected nodes to reduce false positive.

Keywords: *NetFlow Protocol, C&C Botnet, Hierarchical Clustering, Correlation, Anomaly Detection, Network Security.*

1. Introduction

The term “Bot” is derived from the word “Robot” that is an intelligent software application that runs via worms, trojans or other malicious codes to perform a group of cyber operations over the Internet. The other name of bot is zombie. A large number of bots form a connected group called a “Botnet” that under the remote control of a human operator called a “Botmaster”.

According to the explanation in [1], botnets can be classified to four categories: IRC, HTTP, P2P and DNS. The IRC bot is a centralized topology that a central point sends messages between clients [2]. In connection phase, the bot program establishes an IRC channel, and connects the zombie to the command and control (C&C) server. Upon establishment of C&C channel, the zombie becomes a part of attacker’s botnet army. Then, the actual botnet command and control activities will be started. The botmaster uses the IRC channel to disseminate commands to his bot army. Bots receive and execute commands sent by botmaster. The IRC channel enables the botmaster to remotely control the action of large number of bots to conduct various illicit activities [3].

In the http bot, the command and control (C&C) is http based [1]. In HTTP-based botnets, bots contact C&C server periodically to fetch commands. The HTTP protocol is a popular communication method by botnet which is difficult to be detected. Botnets usually bypass security devices using the HTTP protocol [4].

Centralized control of botnets offers a single point of failure for the botnet. So, more stable architectures will be used by botnet operators, P2P architecture [1]. P2P communication system has several important advantages over centralized networks. It is much harder to disrupt and the design of P2P systems are more complex. This means that the compromise of a single bot does not necessarily mean the loss of the entire botnet. But, there are typically no guarantees on message delivery or latency [2].

When bots used DNS as a communication channel to connect to botmaster, called the DNS botnets. Recent DNS botnets are so difficult to be detected, because using new fluxing features. Fast flux (FF), a mechanism that a set of IP addresses change frequently corresponding to a unique domain name. Domain flux (DF), a mechanism that a set of domain names are generated automatically and periodically corresponding to a unique IP address [5].

Botnet detection and defense is an important research task. There are mainly many categories of botnet detection and tracking methods. The detection techniques are classified into four classes in [3]: signature-based, anomaly-based, Domain Name System (DNS)-based and mining-based techniques.

According to [4], there are two approaches of botnet detection, one is signatures based method and the other is based on anomaly. In [6], detection methods are traffic based, anomaly based, nick name based, independent of C&C controls and spam mail based.

In this research, the main data-source for botnet detection is NetFlow records. NetFlow protocol has the task of network traffic analysis and plays a vital role in network troubleshooting, performance improvement and increased availability of members. The protocol stores information about network nature, such as who, when and how has used network traffic. In last, the SNMP protocol used for monitoring network traffic. SNMP however aggregates information on a very high level (i.e. network interface throughput or device uptime), but, a lot of information is lost. The SNMP protocol can't specify that who and how much has used bandwidth. So, the NetFlow protocol designed for collects information of network layer [7].

A flow is a unidirectional stream of packets that pass through a network element and share a common set of attributes [8]. NetFlow version 9 is the latest release from Cisco and was also selected by the IETF as the basis for the IPFIX standard. NetFlow protocol is useful in identification of agents producing an additional load to the network. Therefore it is very effective in identifying unusual programs such as botnet.

Several mechanisms have been proposed for botnet detection; One of these mechanisms, uses NetFlow protocol. Low volume of data, simplicity of computation, lower false positive and being online are some of the advantages of this mechanism against other mechanisms. Network flow records high-level descriptions of Internet connections, but not the actual data transferred [9].

Various methods are used to analyze NetFlow data. One of these methods is clustering. Bots have a specific traffic pattern to communicate over the network, like ports, size of the packets in each direction, and duration. So, if various data mining techniques combined with clustering analysis, specific patterns of behavior in network traffic are identified and can be found bots.

We proposed a method for C&C bot detection that contains 5 steps: First, we collect network traffic and events. Second, records refine based on basic filtering, white list and black list. Third, we use clustering technique to create two clusters of remaining flows and events. Forth, we refine the flows cluster based on patterns, policies, and events cluster. In last step, we use rules and pervious information to discover correlations and certain patterns, and report the bot cluster(s).

The remainder of this paper is structured as follows. Section 2 discusses on related work, defines the scope of this article and highlights innovative aspects of our approach. In section 3, the general methodology as well as proposed approach is described. Finally, we offer the simulation of our approach in section 4 and conclude in section 5.

2. Related Work

Botnet detection and tracking has been a major research topic in recent years. Different solutions have been proposed in different sources. Traditionally, botnets have mainly been identified in several ways. According to the explanation in [3], botnet detection techniques can be classified as being signature-based, anomaly-based, DNS-based, and mining-based. Several of the prevalent data types that used for botnet detection and tracking have DNS data, NetFlow data, packet tap data, address allocation data, honeypot data and host data [2].

According to [10], bot detection mechanisms contain infiltration, C&C server hijack, syntactic, horizontal correlation, vertical correlation, host-based and network-based. An addition, botmaster detection mechanisms contain marking, logging and stepping-stone detection that explained in [10].

In this paper, data source is NetFlow data. NetFlow is a traffic profile monitoring technology developed by Darren Kerr and Barry Bruins at Cisco Systems, back in 1996. NetFlow data provides important information about network conversations and behaviors. Each unique flow is recorded by the network devices or probes, and the flows are then reported to a data collection server [11]. NetFlow can help network managers to monitor suspect Internet botnet's activities by analyzing the source data from the router [12]. NetFlow data represents information gathered from the network by sampling traffic flows and obtaining information regarding source and destination IP addresses and port numbers [13].

A survey that aims at analyzing, classifying and comparing the most relevant network-based botnet detection methods, can see in [14]. Generally, botnet detection methods using NetFlow are classified in five categories: correlation, clustering, IRC, machine learning and DNS.

Correlation: Disclosure, a large-scale, wide-area botnet detection system incorporates a combination of novel techniques to overcome the challenges imposed by the use of NetFlow data. Several groups of features allow Disclosure to reliably distinguish C&C channels from benign traffic using NetFlow records [15]. BotHunter that explained in [16], is analytical strategy of matching the dialog flows between internal assets and the broader Internet as dialog-based correlation, and contrast this strategy to other intrusion detection and alert correlation methods. A same approach presented for C&C bot, in [17].

Clustering: in this approach, a clustering scheme with a set of traffic features design to determine group similar traffic patterns. BotMiner defines a botnet as a coordinated group of malware instances that are controlled via C&C communication channels. In BotMiner, detection framework clusters similar communication traffic and similar malicious traffic, and performs cross cluster correlation to identify the hosts that share both similar communication patterns and similar malicious activity patterns [18]. BotTrack is an approach based on clustering and PageRank algorithm that analyzes communication behavioral patterns and to infer potential botnet activities [19]. An approach proposed based on PageRank and MapReduce in [20].

IRC: Internet Relay Chat (IRC) is a concept that allows users to communicate with each other in real time. There exist several separate networks of so called IRC servers, which provide users with a connection to IRC. A common method of an attacker to communicate with the botnet is to use IRC. Rishi is an approach to detect IRC characteristics of infected machines that explained in [21]. There are more methods for detect IRC bot that explained in [22], [23], [24] and [25].

Machine Learning: Reference [26] uses machine learning techniques to identify the command and control traffic of IRC-based botnets.

DNS: A survey presented in [5], aims to classify botnet detection methods. There is a method that uses DNS data and traffic network to detect malicious hosts that explained in [27].

In summary, the automatic detection of botnets is a challenge and demands for a different approach. We fill this gap by designing a method to detect C&C channels based on traffic analysis and data mining techniques.

Proposed approach uses clustering technique to create two clusters of flows and events. In addition, it uses of a correlation engine to combine clusters information. The result cluster refines based on rules and pervious information to specify bot cluster(s). The approach uses clustering and correlation benefits that proposed in other papers. In addition, this approach adds a rule-based logic and information combination to reduce false positive.

3. Proposed Approach

In this section, we discuss on proposed approach. First, we present basic definitions. Second, we offer network configuration for collect flows and events. Third, we express proposed approach and explain different parts of the algorithm.

3.1 Basic Definitions

Definition1. A C&C botnet uses a protocol to communicate instructions and reports, signal a bot's state and availability as well as transmit bot program updates between one or more bots and the controlling entity, the bot master [28].

Definition2. A flow is a unidirectional stream of packets that pass through a network element and share a common set of attributes [8].

Definition3. Two flows are said to be correlated when they exhibit one or more common properties [22]:

- They are the product of similar applications.
- There is a causal relationship.
- There is one transmitter and multiple receivers.

Definition4. Types of Correlation [15]:

- Vertical Correlation: detecting command and control (C&C) channels used by botmasters to communicate with each infected machine.
- Horizontal Correlation: botnet detection is based upon patterns of crowd behavior exhibited by collections of bots in response to botmaster commands.

3.2 Flows and Events Collectors

The sensor is the device or program that captures data from your network and forwards it to the collector. The collector is software that receives sensor records and writes them to disk. For collecting flows and events, we must configure two collectors on network. Flow Collector saves flow records that capture from the network. Event Collector saves alert records that reported by security sensor, such as IDS, Firewall, Antivirus and etc. Figure (1) shows configured network that collects flows and events and we will describe in continue. Also, figure (2) illustrates our methodology for C&C botnets detection that will explain in section 3.3.

3.2.1 Flow Collector

The biggest flow collectors on Internet are cflowd, flowd and flow-tools. Flow-tools, the most commonly used flow management tool kit. To install the latest version of flow-tools from source, download the source code from http://code.google.com/p/flow-tools/, and extract it. Now go into the flow-tools directory, and read the install file for the current compiling instructions. The process probably includes the steps configure, make, and make install [29].

The flow-capture program listens on a specified UDP port for incoming flow exports. It then captures the data and writes flow records to disk. It configure on a special IP and port. Many network hardware manufacturers, such as Cisco, include flow export in their products. Configure NetFlow on a Cisco router interface.

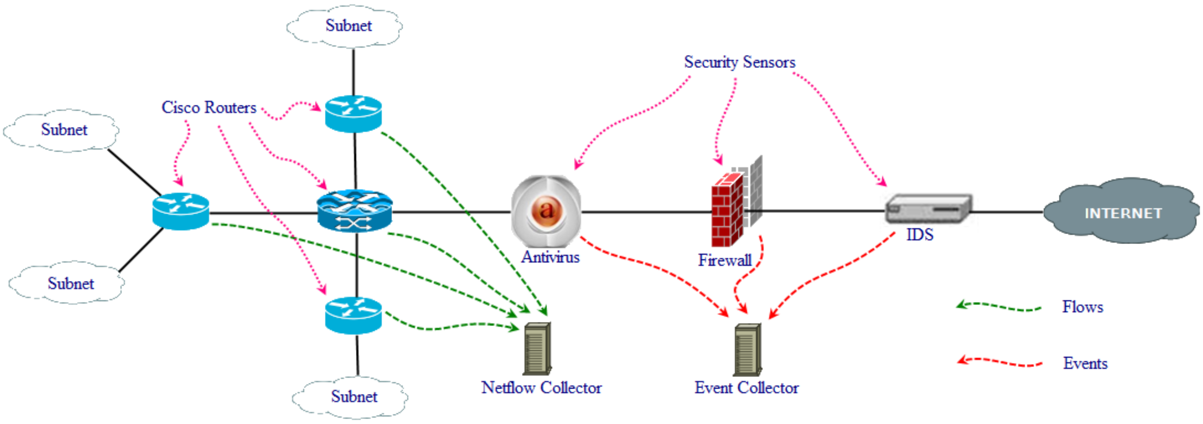


Fig. 1 Network configuration

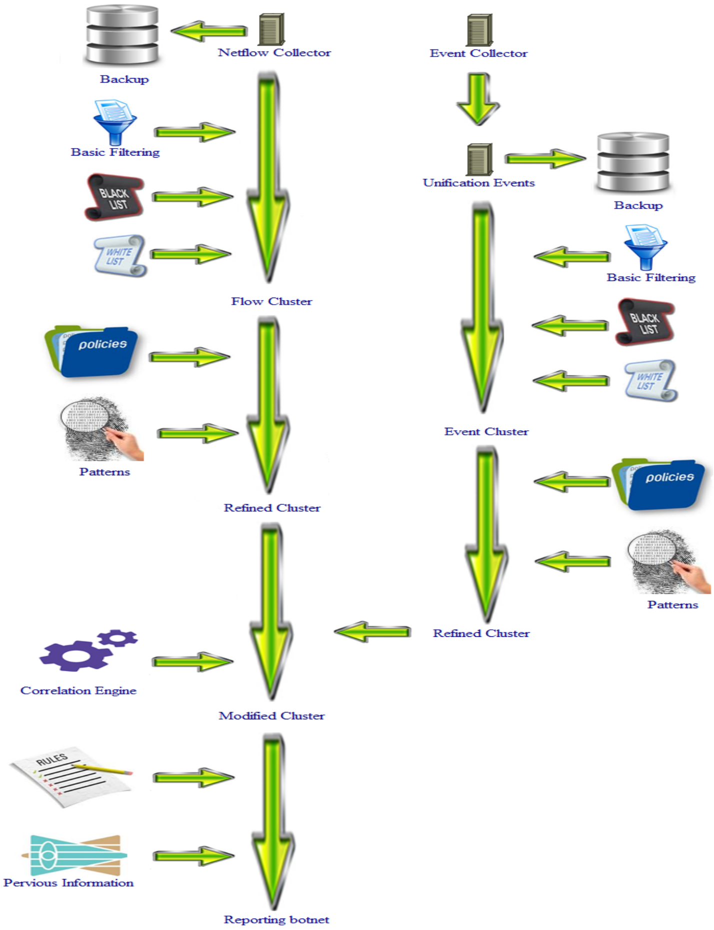


Fig. 2 An anomaly methodology for detect C&C botnets

3.2.2 Event Collector

All the network events from security sensors sent at a central location. Events collector allows us to get events from remote sensors and store them in a local event log on the collector computer. All data in the forwarded event is saved in the Event Collector. Open Source Security Information Management (OSSIM) is an open source security information and event management system, integrating a selection of tools designed to aid network administrators in computer security, intrusion detection and prevention. We configure security sensors on OSSIM, such that, sensors sent events to interface that connected to OSSIM.

3.3 Methodology

In this section, we explain a new botnet detection anomaly approach that the goal detects same patterns belong to the same botnet in network. Botnets have similar behaviors during a time period. Our approach finds similarity behavior of hosts using NetFlow information and reported events through a predefined time period.

We proposed an anomaly methodology for C&C bot detection that contains below steps:

1. Collecting network traffic and events
2. Refining records based on basic filtering, white list and black list
3. Creating two clusters of remaining flows and events
4. Refining clusters based on patterns and policies, and merging clusters information together
5. Using rules and pervious information to discover correlations and certain patterns, and reporting the bot cluster(s)

In first step, we collect NetFlow data and events of sensors and store in Flow and Event Collectors that explained in section 3.2.

In next step, we refine NetFlow and event records based on basic filtering, black list and white list. Before this, we must unify events. Unification Events level aims to unifying events from all security sensors in a single format on just one console. We will be able to observe all security events for a particular moment in time on the same screen and in the same format. So, we will need to organize a translator and a database. The translator unifies all network events that coming from different sensors, in same format and stores them in the same database.

In the basic filtering level, we filter out all the flows that are not directed from internal hosts to external hosts. Therefore, we ignore the flows related to communications between internal hosts and flows initiated from external hosts towards internal hosts. We also filter out flows that are not completely established, i.e., these flows are mainly caused by scanning activity (e.g., when a host sends SYN packets without completing the TCP hand-shake) [18]. In white list filtering, we filter out those flows whose destinations are well known as legitimate servers (e.g., Google, Yahoo!) that will unlikely host botnet C&C servers. In black list filtering, we filter out those flows whose destinations are well known as malicious servers.

In third step, we create two clusters of remaining flows and events. Clustering technique enables us to identify groups of similar C&C flows and events in our data set. We use clustering for three main reasons. Often, the message lengths of the messages in a C&C flow are not equally static throughout several C&C flows of one botnet, but show slight deviations in a small range. Thus, we need to aggregate similar flows and learn the range of each message's length in a C&C flow. Second, grouping similar C&C flows into clusters allows us to build a centroid for each cluster which represents the fingerprint of this cluster's C&C flows. Third, the similar behaviors of same botnets make same events groups in regular periods. The clustering step produces efficient representations that serve as training data for the subsequent classification of flows. In addition, the clustering results provide insights into and measure the relationships between clusters of different malware families [28]. We need to select a clustering method that has above specification. This definition of similarity between flow and event groups gives us the opportunity to apply hierarchical clustering. This allows us to build a dendrogram, i.e., a tree like graph (see Fig. 3) that encodes the relationships among the bots.

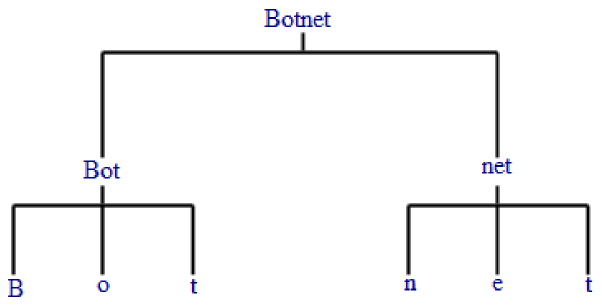


Fig. 3 Example of hierarchical clustering for botnet detection

Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types:

- Bottom-up approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.

- Top-down approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy

There are other clustering methods that used in [30], [31] and [32]. Reference [30] uses the spectral clustering method. Of course, references [18], and [28] use hierarchical clustering.

In next step, we refine clusters based on patterns and policies. The clusters combine their information. Patterns are initial signs of a possible attack. In this level, we attempt to determine behavior patterns of C&C botnets that help us identify botmaster, malicious hosts, path taken, and attack behavior and method. We define attack behavior as the sequence of attacks and activities carried out by a botmaster on one or more malicious machines. Policy are sets of conditions, constraints, and settings that allow us to designate who is authorized to connect to the network and the circumstances under which they can or cannot connect. We must survey flow and event clusters to see that flows are based on policies and events ignore, which policies. In last level of this step, we merge two cluster based on a correlation engine. The correlation engine executes an operation on input clusters and return output cluster. Correlation in our methodology means the ability to view all flows and events in all systems in one cluster and in the same format to improve detection capabilities. Correlation improves reliability, sensitivity, and abstraction.

In final step, we use rules and pervious information to discover correlations and certain patterns, and report the bot cluster(s). We define lists of rules for each sequence of correlation flows that we expect as behavior C&C botnets. The rules aimed at locating unknown or undetectable attacks, since patterns that characterize them are unavailable. In last level, we combine the new and old information to reduce false positive.

The approach, as an integrated solution, offers monitoring at any level, from the lowest to the highest. We proposed an anomaly detector that the detection does in multi stages:

- Basic filtering, black list and white list

- Patterns and polices

- Correlation engine

- Rules and pervious information

4. Simulation

In this section, we evaluate our approach. For this end, we simulate a network by GNS3. We configure a network on GNS3 that infected to Zeus bot. Zeus is computer malware that used to steal banking information by man-in-the-browser, keystroke logging and form grabbing. In this paper, we use centralized Zeus botnets that all bots only connect to a botmaster and receive command and control traffic from only a server. Flow characteristics that we need to detect Zeus botnet, fall into two categories:

- Static characteristics: source and destination IP address, source and destination port numbers and protocol
- Dynamic characteristics: packet event times, bytes per packet, periodic throughput samples, post and get

We need post and get, because Zeus uses HTTP. We show configured network on GNS3 in Fig. 4. We send a copy of network traffic to OSSIM on the interface of Cisco router that IP has 192.168.100.1. OSSIM can collect flows and events and stores in database. We setup IDS on OSSIM. So, we have a flow database and a unification event database. Figure (5) shows part of the network traffic in a special time period. For next step, we ignore black list and white list levels. In basic filtering, we filter out the flows related to communications between internal hosts and flows initiated from external hosts towards internal hosts.

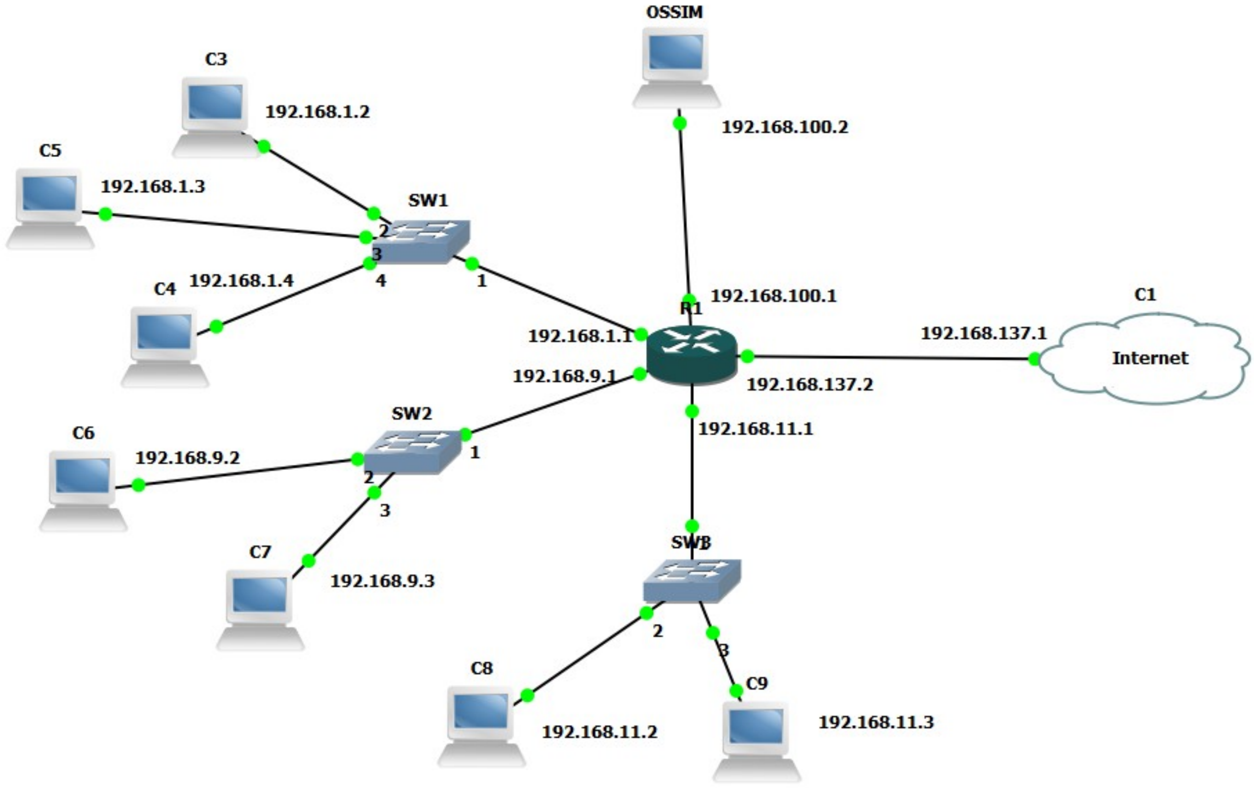


Fig. 4 Network simulation on GNS3

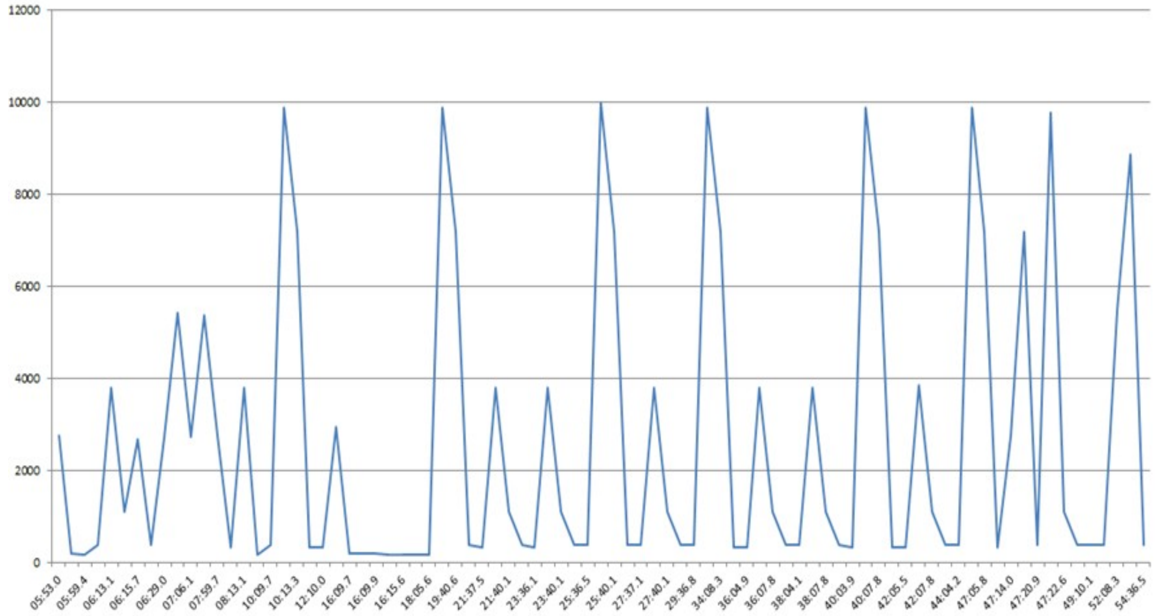
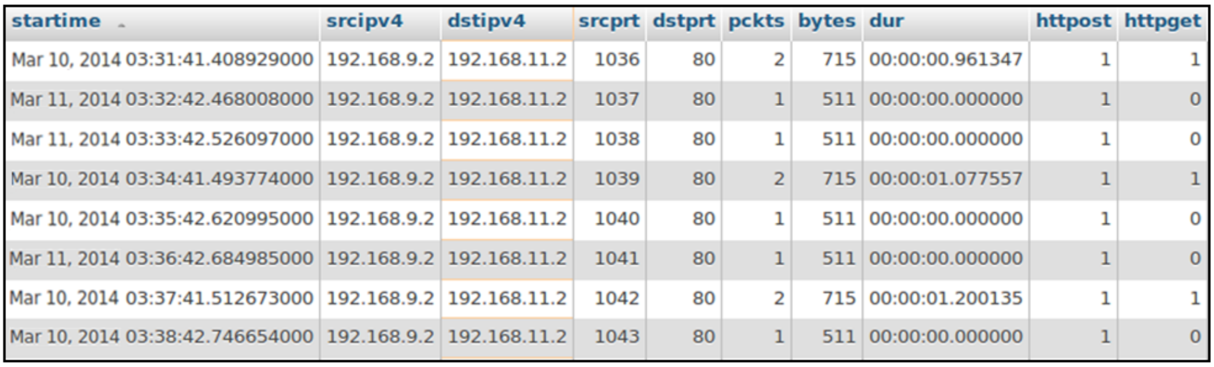


Fig. 5 Part of the network traffic

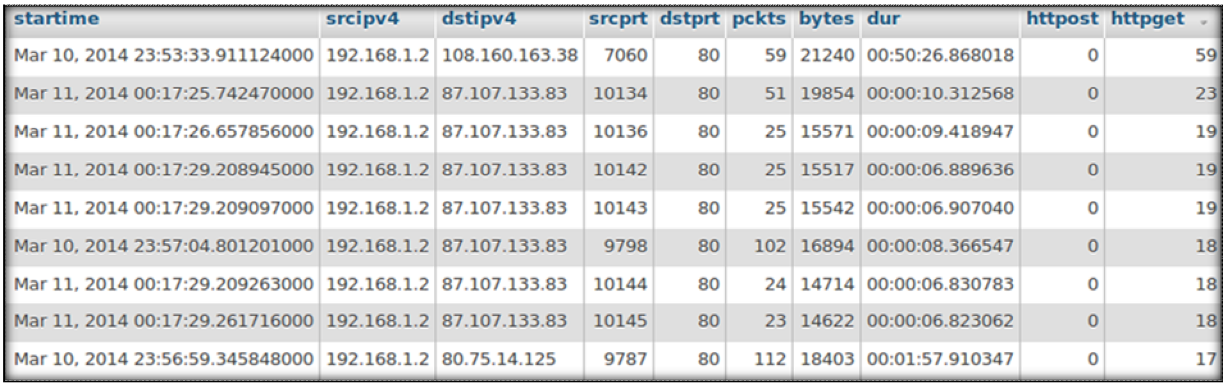
Various botnets have different behavior patterns on network. We create flow cluster based on number of packets, bytes per packet, duration, get and post, because Zeus is an unknown bot for me, before we detect it and understand its behavior patterns. Also, we do not policy level, because have no policy for the network.

Events cluster can't help us, because Zeus uses HTTP. We modified the flow cluster in correlation engine and go to next step. In rules level, according to Fig. 5, rules understand that a special pattern exists in traffic. We have two important clusters that have similar specification. Some flow records of this clusters show in fig. 6 and fig. 7.



startime	srcipv4	dstipv4	srcprt	dstprt	pckts	bytes	dur	httpost	httpget
Mar 10, 2014 03:31:41.408929000	192.168.9.2	192.168.11.2	1036	80	2	715	00:00:00.961347	1	1
Mar 11, 2014 03:32:42.468008000	192.168.9.2	192.168.11.2	1037	80	1	511	00:00:00.000000	1	0
Mar 11, 2014 03:33:42.526097000	192.168.9.2	192.168.11.2	1038	80	1	511	00:00:00.000000	1	0
Mar 10, 2014 03:34:41.493774000	192.168.9.2	192.168.11.2	1039	80	2	715	00:00:01.077557	1	1
Mar 10, 2014 03:35:42.620995000	192.168.9.2	192.168.11.2	1040	80	1	511	00:00:00.000000	1	0
Mar 11, 2014 03:36:42.684985000	192.168.9.2	192.168.11.2	1041	80	1	511	00:00:00.000000	1	0
Mar 10, 2014 03:37:41.512673000	192.168.9.2	192.168.11.2	1042	80	2	715	00:00:01.200135	1	1
Mar 10, 2014 03:38:42.746654000	192.168.9.2	192.168.11.2	1043	80	1	511	00:00:00.000000	1	0

Fig. 6 Some flow records of cluster 1



startime	srcipv4	dstipv4	srcprt	dstprt	pckts	bytes	dur	httpost	httpget
Mar 10, 2014 23:53:33.911124000	192.168.1.2	108.160.163.38	7060	80	59	21240	00:50:26.868018	0	59
Mar 11, 2014 00:17:25.742470000	192.168.1.2	87.107.133.83	10134	80	51	19854	00:00:10.312568	0	23
Mar 11, 2014 00:17:26.657856000	192.168.1.2	87.107.133.83	10136	80	25	15571	00:00:09.418947	0	19
Mar 11, 2014 00:17:29.208945000	192.168.1.2	87.107.133.83	10142	80	25	15517	00:00:06.889636	0	19
Mar 11, 2014 00:17:29.209097000	192.168.1.2	87.107.133.83	10143	80	25	15542	00:00:06.907040	0	19
Mar 10, 2014 23:57:04.801201000	192.168.1.2	87.107.133.83	9798	80	102	16894	00:00:08.366547	0	18
Mar 11, 2014 00:17:29.209263000	192.168.1.2	87.107.133.83	10144	80	24	14714	00:00:06.830783	0	18
Mar 11, 2014 00:17:29.261716000	192.168.1.2	87.107.133.83	10145	80	23	14622	00:00:06.823062	0	18
Mar 10, 2014 23:56:59.345848000	192.168.1.2	80.75.14.125	9787	80	112	18403	00:01:57.910347	0	17

Fig. 7 Some flow records of cluster 2

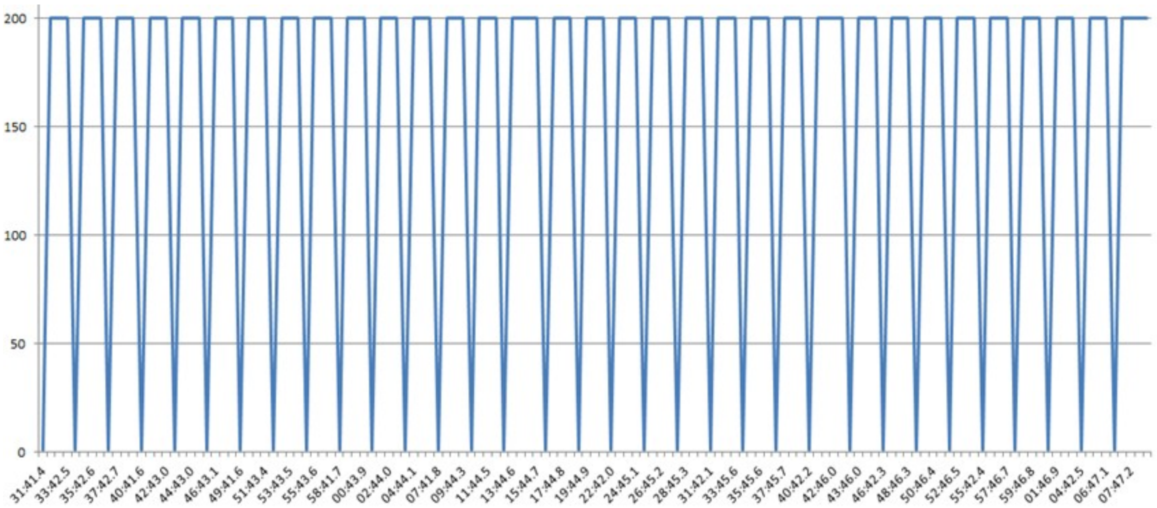


Fig. 8 Behavior pattern of Zeus bot

You can see get and post pattern of Zeus bot in fig. 8. Bots usually connect to botmaster for a few seconds in special periods. Bots have same behavior in different periods. We can see records with same specification in cluster 1. According to what we know about botnets, define rules. So, rules level reports cluster 1 as bot cluster, because we predefine this specification as bot in rules level.

5. Conclusions

In this paper, we have shown a methodology for anomaly detection of C&C botnets that incorporates a combination of novel techniques to overcome the challenges imposed by the use of NetFlow data and events. The approach detects botnets in multi stages, so we expect to reduce false positive. Bots have behavior on network that they distinguish from normal traffic. So, we can create number of hierarchical clusters of network flows and events and find similar behavior and detect botnet cluster(s) based on basic filtering, black list and white list, patterns, polices, correlation engine, rules and pervious information.

References

[1] Z. Zhaosheng, L. Y. C. Guohan, F. Zhi, P. Roberts, and H. Keesook, “Botnet Research Survey”, 32nd Annual IEEE International Computer Software and Applications, Turku, July 2008, pp. 967-972.

[2] M. Bailey, E. Cooke, F. Jahanian, X. Yunjing, and M. Karir, “A Survey of Botnet Technology and Defenses”, Conference For Homeland Security, Cybersecurity Applications & Technology, Washington, March 2009, pp. 299-304.

[3] M. Feily, A. Shahrestani, and S. Ramadass, “A Survey of Botnet and Botnet Detection”, Third International Conference on Emerging Security Information, Systems and Technologies, Athens, June 2009, pp. 268-273.

[4] C. Li, W. Jiang, and X. Zou, “Botnet: Survey and Case Study”, 4th International Conference on Innovative

Computing, Information and Control, Kaohsiung, December 2009, pp. 1184-1187.

[5] Z. Lei, Y. Shui, W. Di, and P. Watters, "A Survey on Latest Botnet Attack and Defense", 10th International Conference on Trust, Security and Privacy in Computing and Communications, Changsha, November 2011, pp. 53-60.

[6] H. S. Nair, and V. S. E. Ewards, "A Study on Botnet Detection Techniques", International Journal of Scientific and Research Publications, Vol. 2, Issue 4, April 2012.

[7] G. Vliek, "Detecting Spam Machines, a Netfow-Data Based Approach", Faculty of Electrical Engineering, Mathematics and Computer Science, Master of Science, University of Twente, Netherlands, February 2009.

[8] I. Drago, R. R. R. Barbosa, R. Sadre, A. Pras, and J. Schönwälder, "Report of the Second Workshop on the Usage of NetFlow/IPFIX in Network Management", Journal of Network and Systems Management, springer, Vol. 19, Issue 2, June 2011, pp. 298-304.

[9] B. Li, J. Springer, G. Bebis, and M. H. Gunes, "A Survey of Network Flow Application", Journal of Network and Computer Application, 2013, pp. 567-581.

[10] S. Khattak, N. Ramay, K. Khan, A. Syed, and S. Khayam, "A Taxonomy of Botnet Behavior, Detection, and Defense", Journal of Communications Surveys & Tutorials, IEEE, Vol. PP, Issue 99, October 2013, pp. 1-27.

[11] S. Choudhary, and B. Srinivasan, "Usage of Netflow in Security and Monitoring of Computer Networks", International Journal of Computer Applications, Vol. 68, No.24, April 2013.

[12] V. M. Dhamdhere, and G. A. Patil, "Netflow Method Used for Internet Worm Detection", International Journal of Scientific & Engineering Research, Vol. 3, Issue 3, March 2012.

[13] M. Lee, N. Duffield, and R. R. Kompella, "Two Samples are Enough: Opportunistic Flow-level Latency Estimation Using NetFlow", 29th Conference on Information, IEEE Press Piscataway, 2010, pp. 2196-2204.

[14] S. Garcia, A. Zunino, and M. Campo, "Survey on Network-Based Botnet Detection Methods", International Journal of Security and Communication Networks, 2013.

[15] L. Bilge, D. Balzarroti, W. Robertson, E. Kirda, and C. Kruegle, "DISCLOSURE: Detecting Botnet Command and Control Servers Through Large-Scale NetFlow Analysis", 28th Annual Computer Security Applications Conference, New York, 2012, pp. 129-138.

[16] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation", Symposium on USENIX Security, Berkeley, Article No. 12, 2007.

[17] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection", 17th Conference on Security Symposium, Berkeley, 2008, pp. 139-154.

[18] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic", 2008.

[19] J. Francois, S. Wang, R. State, and T. Engel, "BotTrack: Tracking Botnets using NetFlow and PageRank", 10th International IFIP TC 6 Conference on Networking, Berlin, 2011, pp. 1-14.

[20] J. Francois, S. Wang, W. Bronzi, R. State, and T. Engle, "BotCloud: Detecting Botnets using MapReduce", IEEE International Workshop on Information Forensics and Security, Iguacu Falls, December 2011, pp. 1-6.

[21] J. Goebel, and T. Holz, "Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation", First Conference on First Workshop on Hot Topics in Understanding Botnets, Berkeley, 2007.

[22] W. T. Strayer, D. Lapsely, R. Walsh, and C. Livadas, "Botnet Detection Based on Network Behavior", Botnet Detect, Springer, 2008, pp. 1-24.

[23] M. Thapliyal, A. Bijalwan, N. Garg, and E. S. Pilli, "A Generic Process Model for Botnet Forensic Analysis", Conference on Advances in Communication and Control Systems, 2013.

[24] W. T. Strayer, R. Walsh, C. Livadas, and D. Lapsley, "Detecting Botnets with Tight Command and Control", 31st IEEE Conference on Local Computer Networks, Tampa, November 2006, pp. 195-202.

[25] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-Scale Botnet Detection and Characterization", First Conference on First Workshop on Hot Topics in Understanding Botnets, Berkeley, 2007.

[26] Livadas, C., Walsh B., Lapsley, D., and Strayer, T., "Using Machine Learning Techniques to Identify Botnet Traffic", 31st IEEE Conference on Local Computer Networks, Tampa, November 2006, pp. 967-974.

[27] S. Arshad, M. Abbaspour, M. Kharrazi, and H. Sanatkar, "An Anomaly-based Botnet Detection Approach for Identifying Stealthy Botnets", IEEE International Conference on Computer Applications and Industrial Electronics, Penang, December 2011, pp. 564 - 569.

[28] C. J. Dietrich, C. Rossow, and N. Pohlmann, "CoCoSpot: Clustering and Recognizing Botnet Command and Control Channels using Traffic

Analysis", Computer Networks, Vol. 57, Issue 2, February 2013, pp. 475–486.

[29] M. W. Lucas, Network Flow Analysis, San Francisco, No Starch Press, 2010.

[30] T. W. Chiou, S. C. Tsai, and Y. B. Lin, "Network Security Management with Traffic Pattern Clustering", Soft computing, Springer, January 2014.

[31] S. Garg, A. K. Sarje, and S. K. Peddoju, "Improved Detection of P2P Botnets through Network Behavior Analysis", Recent Trends in Computer Networks and Distributed Systems Security Communications in Computer and Information Science, Vol. 420, 2014, pp. 334-345.

[32] K. Muthumanickam, E. Ilavarasan, and S. K. Dwivedi, "A Dynamic Botnet Detection Model based on Behavior Analysis", International Journal on Recent Trends in Engineering & Technology, Vol. 10, Issue 1, January 2014, pp. 104-111.