Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 14, no. 5, May 2018, pp. 1050-1059
DOI: 10.23940/ijpe.18.05.p24.10501059

BotCapturer: Detecting Botnets based on Two-Layered Analysis with Graph Anomaly Detection and Network Traffic Clustering

Wei Wang[a,b,*], Yang Wang[a], Xinlu Tan[a], Ya Liu[a], and Shuangmao Yang[b]

[a]*Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing, 100044, China*
[b]*Science and Technology on Electronic Information Control Laboratory, Chengdu, 610036, China*

Abstract

Botnets have become one of the most serious threats on the Internet. On the platform of botnets, attackers conduct series of malicious activities such as distributed denial-of-service (DDoS) or virtual currencies mining. Network traffic has been widely used as the data source for the detection of botnets. However, there are two main issues on the detection of botnets with network traffic. First, many traditional filtering methods such as whitelisting are not able to process the very large amount of traffic data in real-time due to their limited computational capability. Second, many existing detection methods, based on network traffic clustering, result in high false positive rates. In this work, we are motivated to resolve the above two issues by proposing a lightweight botnet detection system called *BotCapturer*, based on two-layered analysis with anomaly detection in graph and network communication traffic clustering. First, we identify anomalous nodes that correspond to C&C (Control and Command) servers with anomaly scores in a graph abstracted from the network traffic. Second, we take advantage of clustering algorithms to check whether the nodes interacting with an anomalous node share similar communication pattern. In order to minimize irrelevant traffic, we propose a traffic reduction method to reduce more than 85% background traffic. The reduction is conducted by filtering the packets that are unrelated to the hosts like C&C server. We collect a very big dataset by simulating five different botnets and mixing the collected traffic with background traffic obtained from ISP. Extensive experiments are conducted and evaluation results based on our own dataset show that BotCapturer reduces more than 85% input raw packet traces and achieves a high detection rate (100%) with a low false positive rate (0.01%), demonstrating that it is very effective and efficient in detecting latest botnets.

Keywords: botnet detection; anomaly detection; network flows; intrusion detection

(Submitted on February 5, 2018; Revised on March 18, 2018; Accepted on April 26, 2018)

1. Introduction

Botnets have been one of the most significant threats of the Internet. Based on the statistics provided by Spamhaus [20], more than 9,500 new botnet C&C servers were active in 2017, which is 32% more than in 2016. A botnet is a network of computers infected and often controlled by malicious software. Attackers can operate botnets to perform various malicious behaviors such as spam, virtual currencies mining, DDoS or identity theft. Botnets are structurally divided into centralized and distributed. Centralized botnets typically have a C&C server through which the master sends control commands to individual bots and then controls the botnet. On the contrary, in a distributed botnet, each bot shares the botnet's information partly and communicates with peers. Botnets usually use IRC, HTTP or TELNET as communication protocols. In IRC-based botnets, all bots and master are in the same channel. But, the number of IRC-based botnets has decreased in recent years since the channel is easy to detect. Thus, the protocols used for C&C channels evolve from IRC to others, especially to HTTP that is very popular among attackers.

Many approaches have been proposed to detect the botnets. The techniques can be classified into signature-based detection and behavior-based detection. For signature-based detection, Gianvecchio et al. [6] detected IRC bots by inspecting the interval, length, frequency, and payload entropy of chat messages. Zhu et al. [19] proposed detecting bot

* Corresponding author.
E-mail address: wangwei1@bjtu.edu.cn

activities based on application-level protocol failures. Zhao et al. [18] detected botnet activities based on traffic behavior analysis by classifying network traffic using machine learning. In general, signature-based bot detection techniques are accurate. But they can be ineffective when bot variants are encountered. As for behavior-based detection, AsSadhan et al. [1] detected botnet by finding the periodic behavior from network traffic. The advantage of such technique is that it is based on a basic property shared by many botnet variants and is independent of the structure (e.g., centralized, P2P) and C&C protocol (e.g., IRC, HTTP). But the bot master can hide the periodic behavior by uniformly randomizing the period. Choi et al. [4] detected botnet activities by monitoring group activities in DNS traffic, assuming that a bot must look up the domain names of the C&C server. Another representative approach is Botminer [1] that performed across clusters of communication traffic and malicious activity. Botminer is independent of botnet structure and C&C protocol. Compared with signature-based detection, behavior-based systems focus on identifying traffic patterns corresponding to particular bot activities by observing bot behaviors. Though behavior-based detection is relatively lightweight since it does not need to maintain a signature database, existing approaches can still be overwhelmed by the large scale of network traffic. Furthermore, the methods only based on clustering algorithms to recognizing the network traffic patterns are not reliable. They can only identify the hosts that share similar communication patterns; they cannot determine whether these identified hosts belong to a botnet. For a large local area network, especially an enterprise internet, multiple hosts are simultaneously infected once the network is infected by a botnet. For a consistent period of time, the infected hosts communicate with the C&C server with a similar frequency. However, they hardly communicate between each other.

The network traffic between hosts presents graph characteristics based on which we can perform anomaly detection. Anomaly detection has been employed in computer system [7,11,12,13] or in mobile applications [14,15,16]. Anomaly (outlier) detection on graphs, however, is a difficult problem, despite its apparent simplicity [9]. To tackle the outlier detection problem, many techniques have been developed in the past decades. Aggarwal et al. provided [2] first result on the problem of structural outlier detection in massive network streams. Zimek et al. [17] discussed some important aspects of the 'curse of dimensionality' in detail and surveyed specialized algorithms for outlier detection from both categories. Perozzi et al. [10] inferred user preference by the so-called focus attributes through a set of user-provided exemplar nodes, and introduced a novel user-oriented approach for mining attributed graphs. They presented RBRP [10], a fast algorithm for mining distance-based outliers, particularly targeted at high-dimensional data sets. Akoglu, et al. [3] proposed an algorithm called OddBall to detect anomalous nodes in weighted graphs.

In this work, we propose a behavior-based framework BotCapturer based on two-layered analysis with graph anomaly detection and network traffic clustering. BotCapturer can reduce a large amount of traffic and produce not only a higher efficiency but also a higher detection rate and a lower false positive rate. In the first layer, we use the network traffic to build an undirected graph and find all nodes whose egonet[1] fit the botnet's model. These nodes are likely to be the IPs of C&C servers of botnets. For minimizing irrelevant traffic, we filter out all the packets that are irrelevant to these IPs. Experimental results show that BotCapturer can reduce over 85% of background traffic and vastly reduce the traffic needed to analyze for traffic clustering. In contrast, the traditional filtering methods like whitelisting can only reduce a small part of traffic. In the second layer, benefited from the efficient traffic filtering method, Botcapturer can quickly identify the hosts that share similar communication pattern by clustering in network traffic. Finally, like a dual authentication mechanism, Botcapturer conducts two-layered analysis of the results produced by the previous two layers to find the egonet who fits the botnet model and all nodes of the egonet that has similar communication patterns. Then, the egonet can be considered as a botnet.

We make the following contributions.

- We propose BotCapturer to detect centralized botnets in large scale networks. It is independent of protocols and requires no priori- knowledge (e.g., C&C addresses/signatures) of specific botnets. BotCapturer is based on the two-layered analysis of anomaly detection in graph and network traffic clustering, which can vastly improve the detection rate compared with existing methods only based on network traffic clustering.
- We collect a large dataset that consists of various types of botnets (HTTP-based, TELNET-based) for related researches. The data contains traffic of five botnets: Mirai, Zeus, Ares, Athena and BlackEnergy and a very large amount of background traffic.
- We propose an efficient method to filter out most irrelevant network traffic for clustering by filtering the packets that are irrelevant with the hosts like C&C server. Evaluation results based on our dataset show that BotCapturer reduces over 85% input raw packet traces and achieve 100% detection rate with only 0.01% of false positive rate.

The rest of this paper is organized as follows. Section 2 introduces the detecting framework BotCapturer. Section 3 introduces the data. Section 4 provides the experimental results. Section 5 concludes our work.

[1] In social network analysis, the "1-step neighbourhood" of a node is specifically known as its "egonet" [3].

2. The Framework of Botcapturer

Botcapturer conducts two-layered analysis. The first layer is anomaly detection of nodes in graph. The second layer is network traffic clustering for grouping the hosts that share similar communication patterns. We analyze the results provided by the previous two layers to obtain some groups of hosts that are considered as botnets. The framework is shown in Figure 1.

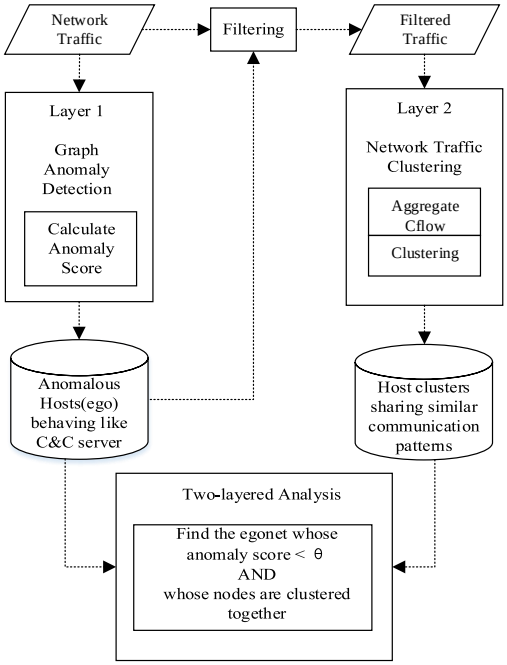


Figure 1. The Framework of Botcapturer

2.1. Anomaly Detection of Graphs

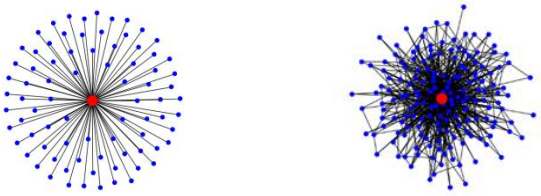
The graph of a typical centralized botnet is a star-shaped structure, where the core node represents the C&C server. Nodes correspond to the bots communicate with the core in familiar frequency. But there is little or no communications between bots because bots hardly connect with each other directly. So, we aim to find a graph structure like Figure 2(a). After detecting, we can find the anomalous nodes that are like the C&C servers of botnets. For these anomalous nodes, the packets whose source IP or destination IP is not anomalous are considered as normal and we can then filter them out. Thus, we can reduce the number of packets required by next layer.

First, abstracting the monitored network traffic into the following triplet, we can obtain the frequency of communication between each pair of IPs. In Equation (1), weight represents the number of communications between IP_1 and IP_2.

$$(IP_1, IP_2, Weight) \tag{1}$$

Mapping the monitored network traffic in undirected graph, each host is considered as a node. The edge represents whether the two nodes communicate with each other, and the weight represents the communication frequency. If there are packets between two hosts, a line can be drawn between the nodes representing the two hosts. The number of packets between them represents the number of communications, and can be used as the weight of the line. This way, we construct an undirected graph that represents the communication of the current network. Intuitively, C&C servers and bots communicate frequently in a botnet. However, bots hardly communicate with each other. In a fixed period of time, suppose that the set of bots in the same botnet $b = \{b_1, b_2, \ldots, b_n\}$, let f_{is} denotes the frequency of communication between b_i and C&C Server S, let f_{js} denotes the frequency of communication between b_j and C&C Server S, $f_{is} \approx f_{js} (1 \le i \le n, 1 \le j \le n)$. However, $f_{ij} \approx 0$. In other words, nodes correspond to bots that have an approximately same weight of connection with

the server and nodes correspond to bots connect to each other with a low weight or having no connection at all. The structure of the botnet is similar to what is shown in Figure 2(a). The center of the diagram represents the C&C server, and the nodes that connect to it represent the bot controlled. In this layer, we aim to find the egonet similar to this structure.



(a) (b)

Figure 2. Illustration of egonet of botnet (a) and egonet of normal (b)

This layer mainly based on the method described in [3] to find anomalous nodes whose egonet conforms botnet's model. For each node (the ego), we consider the induced subgraph of its neighboring nodes as a egonet.

Firstly, extract three features of the graph:

- N_i : number of neighbors (degree) of ego i.
- E_i : number of edges in egonet i.
- w_i : total weight of egonet i.

The law EDLP (Egonet Density Power Law) followed by N_i and E_i in a normal egonet is defined in Equation (2).

$$E_i \propto N_i^{\alpha}, \qquad 1 \leqq \alpha \leqq 2 \tag{2}$$

Then, calculate anomaly score for each ego in the graph. As shown in Equation (3), the score denotes the degree of deviation from the E_i and the theoretical value of edge.

$$anomaly\_score(i) = out\_line(i) + out\_lof(i) \tag{3}$$

We apply this theory to our experiment to find egonets like Botnet, as shown in Figure 2(a). Taking into consideration the characteristics of botnets, we add a value of C.V (Coefficient of Variance) to the anomaly score. As shown in Equation (4), σ is the standard deviation of all weights in egonet, and μ is the average of all weights. The C.V value is mainly used to characterize the similarities between adjacent node weights in egonet.

$$CV = {}^{\sigma}/_{\mu} \tag{4}$$

Since the weights of nodes connected to the C&C server are very similar, the coefficient of Variance of the ego corresponding C&C server of botnet would be very small, min (cv) = 0. The amended formula is shown in Equation (5).

$$anomaly\_score(i) = out\_line(i) + out\_lof(i) + CV(i) \tag{5}$$

In experiments, scores of the egos correspond to the C&C servers of the five botnets ranked the top 10 in 200k IPs. The results of the experiment will be presented in Section 4.

2.2. Network Traffic Clustering

Our clustering model is based on C-flow and its four features [1]. A C-flow is a collection of all flows over a period of time. For clustering, there are four features for each C-flow and can be mapped to a vector whose length is 52, which can describe

the overall distribution of the four features of each C-flow. They are: fph(the number of flows per hour), ppf(the number of packets per flow), bpp(the average number of bytes per packet) and, bps(the average number of bytes per second). The main purpose of clustering is to bring C-flow with similar communication patterns together. For a botnet, the get and push mechanisms between each pair of bot and the C&C server are hardcoded in the program [1]. Depending on the mechanism, bots are pre-programmed to contact each other to update bot's data, receive commands, and send keep-alive messages. Therefore, in the same botnet, the communication patterns between each pair of bot and the C&C server are quite similar. Thus, we can indicate the communication patterns as vectors and then perform the clustering on the vectors. Based on the analysis above, vectors that correspond to bots in the same botnet are supposed to be in the same cluster. Thus, hosts that share the same communication pattern are recognized by clustering.

2.2.1. Filtering Raw Packets

As the centralized botnets matches the structure shown in Figure 2(a), we can only focus on packets whose IPs are contained in star structure. Based on this, we list all anomalous egos with a star structure in the first layer of BotCapturer and filter off packets whose source IP or destination IP is not in the result list. Then we can reduce the background traffic over 85%, which reduces the amount of data needs to be analyzed. What's more, we use whitelisting to filter a portion of the traffic further. We obtain top500 popular website addresses provided by Alexa.com and then gets their server's IP to form our whitelists.

2.2.2. Extension of Flow Concept

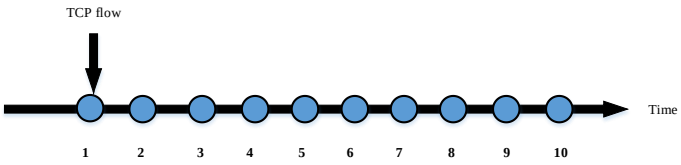
In the experiments, we find that the original definition of flow does not apply to botnets that always communicate through only one TCP flow. For example, in the TELNET-botnet Mirai, the communication between bot and C&C server from beginning to the end has only one TCP flow. For sending keep-alive messages, each bot sends two packets to C&C server in the old TCP flow rather than initiates a new one. However, for one Botnet based on HTTP protocol, as shown in Equation (6), the number of TCP flows has a linear relationship with time.

$$flows = \frac{duration}{interval} \tag{6}$$

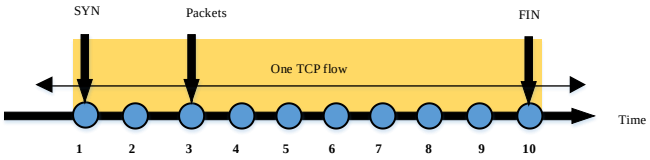
The $interval$ represents the time interval that bot sends keep-alive message to C&C server, and $duration$ represents the running time of the botnet client. As far as the botnet based TELNET is concerned, as shown in Equation (7), the number of TCP flows has nothing to do with time.

$$flows = 1 \tag{7}$$

Suppose that a bot sends keep-alive packets to C&C server every minute, the relationship between the number of TCP flow and time for the two situations are shown in Figure 3.



(a) The botnet based on HTTP



(b) The botnet based on TELNET

Figure 3. The contrast of TCP flows between the botnet based HTTP and the botnet based TELNET

Focusing on the TELNET-based botnets, the vectors of FPH, PPF, BPP and BPS do not denote the communication pattern of a host any longer as the TCP flow is the most basic unit of C-flow. Therefore, we redefine the unit of research as FLOW:

For a TCP flow, suppose it contains a set of $packets = \{p_1, p_2, \dots, p_n\}$ that not contains the SYN and FIN. Take any consecutive even number of packets randomly, for example 8, $p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8 \in packets$, the respective time stamps correspond to each packet are $t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8$. If $\delta = t_5 - t_1 \simeq t_6 - t_2 \simeq t_7 - t_3 \simeq t_8 - t_4$ and $t_5 - t_4 \gg \delta$, then the unit consisted of p_1, p_2, p_3, p_4 is considered a Flow, and the unit consisted of p_5, p_6, p_7, p_8 is considered a different FLOW. Thus, this TCP flow is going to be split up into many FLOWs. The contrast result is shown in Figure 6.

2.2.3. Clustering Method

We employ K-Means [8] for the clustering, as it is a popular clustering algorithm. In K-means, in order to avoid choosing the best value of K, we make an analysis as follows. During clustering, when K increases, the degree of similarity between the samples in the clusters increases too. For a botnet, the bots are grouped together from the beginning because the similarity of communication pattern, where the $occupancy$ is small. When k reaches a certain value k_i, the $occupancy$ rises and exceeds the certain threshold θ. With the continuous increase of k, the $occupancy$ remains unchanged or continues to rise until k exceeds another certain k_j. Then, the clustering granularity becomes so fine that a botnet will be forcibly split into multiple clusters. Equation (8) shows how to get the value of $occupancy$ for one cluster.

$$occupancy = \frac{N_e}{N_c} \quad (0 < occupancy \leq 1) \tag{8}$$

N_e = total number of nodes in an egonet, N_c = number of elements in the cluster

Based on the above analysis, all samples representing bots of botnet are completely assembled in one cluster stably without other samples in the same cluster when $k_i \leq k \leq k_j$. Thus, we can determine whether hosts in an egonet have a similar communication pattern in this way: Determine whether all nodes of an egonet can be completely clustered in one cluster stably, and that there is no other sample in the same cluster with k increases. This way, we get rid of the dependence on optimal k. There are two situations where nodes in an egonet are not clustered into the same cluster, which result in that the occupancy unable to be calculated. The egonet's occupancy has never reached θ, and the egonet's occupancy has reached θ. To describe these two different situations, we set the occupancy to 0 and 2 respectively.

The specific process is as follows:

- The initial value of k is set as 2 and incremented by 1 until $k = n$ or satisfy the end conditions of the experiment.
- When k = 2, if the nodes in an egonet are not clustered into one cluster, set occupancy = 0. Abandon
- When k > 2, if the nodes in an egonet are not clustered into one cluster before the occupancy reaching θ. Abandon
- In case of $\theta \leq \text{occupancy} \leq 1$, if k exceeds k_j, then set the $\text{occupancy} = 2$.
- End the process when all egonets' occupancy = 0 or ≥ 1.

Finally, all hosts of egonets with occupancy > = 1 are considered to have similar communication patterns.

2.3. Two-layered Analysis and Detection

The two-layered analysis and detection takes advantage of the dual authentication mechanism. The first layer verifies whether the anomaly score of the ego is less than the threshold. The second layer verifies whether the communication pattern of all neighbor nodes in egonet is similar. For the first layer, if it considers the ego as anomalous, then go to the second layer. If it also considers the ego as anomalous in the second layer, then we consider the egonet of this ego as a botnet. Finally, using this dual authentication mechanism, we observe the egonet who fits the botnet model and all nodes of it have similar communication patterns. The flow chart that describes whether an egonet is botnet is shown in Figure 4. For any ego named X, suppose the set of one-step neighbors is $\{N_0, N_1, \dots, N_i\}$. Suppose the set of clustering result is C=$\{c_0, c_1, \dots, c_i\}$. If X is a C&C server, the C-flow set $\{Cflow_0, Cflow_1, \dots, Cflow_i\}$ corresponds to all 1-step neighbor nodes $N_0, N_1, \dots, N_i$ of the X, will be clustered in $c_k \in C$, and the c_k only contains these C-flows.

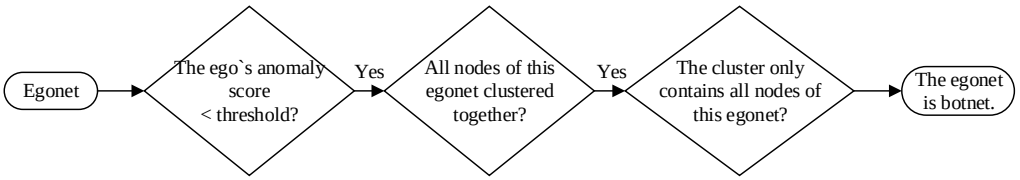


Figure 4. Identify whether an egonet is botnet

3. Data Description

We collected real traffic and have formed our own dataset. We obtained five botnet samples on the Internet as shown in Table 1: Mirai, Zeus, Ares, Athena and BlackEnergy, as these botnets are very representative. Taking advantage of Docker technology [21] makes it easy to simulate a large number of machines on a single physical Machine. We built these botnets and simulated a large number of bots for five botnets respectively in our laboratory. Then, we captured network traffic of each botnet for a couple of days as shown in Table 1. In addition, we captured background traffic from the ISP's gateways for 10 hours.

Since there was no other process running on the simulated hosts except the botnet clients, each virtual machine only had malicious traffic and no normal traffic. However, when a typical host is infected by a botnet client, it has both normal and malicious traffic, and the size of its normal traffic is much larger. To simulate the infected host, we mixed the malicious traffic and the background traffic. We randomly selected the same number of IPs from 200K IPs of the background traffic. Finally, we got a big dataset that contains a lot of background traffic and relatively little botnet traffic. The data is described in Table 1.

Table 1. Collected botnets and captured traffic

Botnet	Protocol	Target	Number of host	Duration	Size	C&C server
Mirai	Telnet	IoT	80	380h	5.1 GB	1
Zeus	Http	PC	80	123h	1.11 GB	1
Ares	Http	IoT	80	63h	2.04 GB	1
Athena	Http	PC	5	90h	3.18 GB	1
Black-Energy	Http	PC	60	140h	0.99 GB	1
Background			203066	10h	743 GB	

4. Experiments and Evaluation

To fully evaluate BotCapturer, we take the first hour of dataset as a small data set for comparing. Thus, we have two datasets, one for the first hour and one for the 10 hours. In order to verify the efficiency of BotCapturer, we aim to detect these five botnets in a large and complex network including C&C channels and each bot.

4.1. Anomaly score of Each Ego

To calculate the anomaly scores for all egos, we filter out those egos that only have one direct neighbor node since there must be at least two neighbor nodes for the ego with star structure. Then, we calculate the anomaly scores for each ego. As shown in Figure 5, the abscissa represents the node ID and the ordinate represents the anomaly score. In Figure 2, (a) and (b) correspond to two extremes of anomaly scores. Score of (a)'s ego is extremely low while (b)'s ego is extremely high. The red dots in Figure 5 correspond to the five C&C servers with low scores.

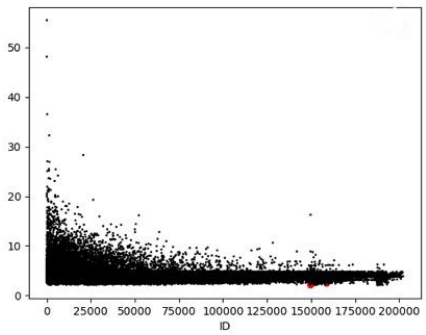


Figure 5. Distribution of anomaly scores of all egos

We set the threshold of the anomaly score as 3.0 reasonably by comparing score of each anomalous egonet. Then, the corresponding raw packets for all nodes above this threshold will be filtered out. Experimental results show that we can filter out over 85% background network traffic. The details are shown in Table 2.

Table 2. Comparison on filtering

Data set	# Packets	# Filtered	Loss
1h	65581004	5373302	91.8%
10h	456229550	70721706	84.5%

4.2. Efficiency validation with flow

Figure 6 illustrates the clustering results (k=200) of 80 hosts of Mirai (TELNET-based). The former is based on flow's original concept where the Mirai's traffic is considered as a continuous flow. In this situation, 80 bots are grouped into one cluster where normal hosts are the majority. For comparison, the latter is based on FLOW where a TCP flow is decomposed reasonably. Correspondingly, the 80 bots and normal hosts are clustered completely into different groups. The results indicate the necessity and accuracy of redefining the flow.

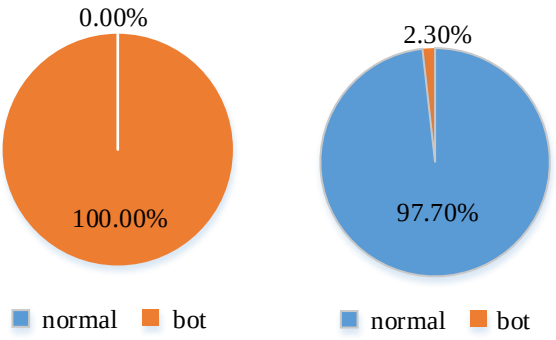


Figure 6. Comparison of Clustering results

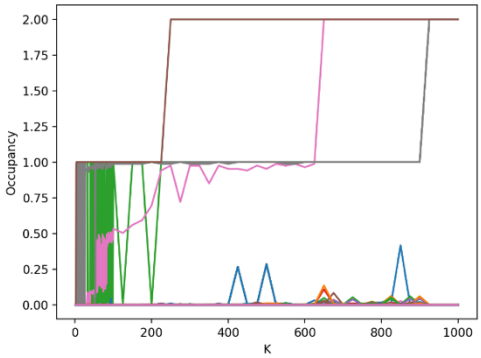


Figure 7. The occupancy of egonets when k increases

4.3. Result of Clustering and Two-layered Analysis and Detection

In a botnet, C-flows, corresponding to each bot, are clustered in a same cluster. However, there is still a small probability that the communication between normal hosts shares a similar pattern with a botnet. Therefore, it is reasonable for the clustering results to contain a small number of normal hosts. The experimental result shows that the occupancy is stabilized within the interval from $\theta = 0.75$ to 1 for a continuous period time when k increases. The occupancy of every ego is shown in Figure 7. It is seen that the number of egonets whose occupancy is not less than θ keeps at a constant value when k increases from 200 to 1000. However, other egonets' occupancies are always less than 0.25. It is reasonable to infer that the nodes in these five egonets have a high similarity in communication patterns. However, although other nodes have not yet satisfied with the condition that makes occupancy to be 0, we can infer that they do not have communication patterns of high similarity because they have not shown obvious characteristics of the strong connections between each other. Thus, although k has not reached n and there are still some egonets whose occupancy are between 0 and θ, the suspicious egonets have been clearly distinguished. Therefore, considering BotCapturer's efficiency, we choose the biggest value before occupancy is set to 2 as the real occupancy for all egonets. The final clustering results are shown in Table 3.

Table 3. Clustering results

Data	Botnet	Protocol	In the Same Cluster	Number of Bots	Clustered Hosts	Occupancy
1h	Mirai	telnet	yes	80	80	100%
	Zeus	http	yes	80	104	76.90%
	Ares	http	yes	80	80	100%
	Athena	http	yes	5	6	83.30%
	Black-Energy	http	yes	60	60	100%
10h	Mirai	telnet	yes	80	80	100%
	Zeus	http	yes	80	83	96.38%
	Ares	http	yes	80	80	100%
	Athena	http	yes	5	5	100%
	Black-Energy	http	yes	60	60	100%

Table 4. Detection results

Data	Number of Egonet	Number of Botnet	TP Rate	FP Rate
1h	5292	5	100%	0
10h	7147	5	100%	0.01%

Finally, BotCapturer performs two-layered analysis using a dual authentication mechanism. The former layer judges whether an egonet conforms the characteristics of a botnet from the graph structure. If so, the second layer verifies whether the communication patterns of each node in the anomalous egonet are similar (that is, whether the communication pattern between each pair of bot and the C&C server is similar). If an egonet is considered as a botnet, all nodes belong to the egonet will be considered as bots. Experimental results show that we successfully identified Botnet's C&C server in a list of all ego's out-of-thresholds. Table 4 shows that BotCapturer can detect centralized botnets with a detection rate of 100% and a false positive rate of 0.01%.

4.4. Limitations

BotCapturer performs well based on two assumptions: 1. botnets are star-shaped; 2. network is stable so that bots can communicate unimpededly with the C&C server in a definite timeframe. However, we cannot assume that all the distributed software behaves like a botnet; this may result in false positives. However, we can avoid this by whitelisting. In addition, with FLOW, BotCapturer can detect centralized or semi-distributed botnets regardless of the C&C protocols. However, it does not work well for P2P types of botnets.

5. Conclusions

The botnets vary and evolve continuously. In this work, focusing on centralized botnets, we propose a framework called BotCapturer based on two-layered analysis of graph and clustering of communication traffic patterns, which is independent of protocols. Benefited from the dual authentication mechanism, BotCapturer vastly improves the detection rates compared with many existing methods that are only based on network traffic clustering. In addition, BotCapturer's filtering method based on the results of anomaly detection in graph reduces over 85% of network traffic which is much more efficient compared with many existing filtering methods that are only based on whitelisting.

We collected a large data set in real computing environments. Extensive experimental results show that BotCapturer achieves 100% detection rate with 0.01% false positive rate on various types of botnets (HTTP-based and TELNET-based). In addition, we redefine a unit referred to Flow as a set of regular packets in the situations where the minimum unit of a conversation is a few of packets rather than a TCP/IP flow in some botnets. The Flow based botnet detection method is proven to be effective to detect some special botnets (e.g., Mirai) that may escape the detection with existing methods.

Our work has not involved the detection of P2P based botnets. In the future work, we are investigating to detect these types of botnets with BotCapturer.

Acknowledgements

We would like to thank Yue Xu and Yaoyao Shang for their help in the data collection in the experiments. The work reported in this paper was supported in part by Funds of Science and Technology on Electronic Information Control Laboratory, under Grant K16GY00040, in part by National Key R&D Program of China, under grant 2017YFB0802805, in part by Shanghai Key Laboratory of Integrated Administration Technologies for Information Security, under Grant GK2015002, in part by the Fundamental Research funds for the central Universities of China, under grant K17JB00060 and K17JB00020, and in part by Natural Science Foundation of China, under Grant U1736114 and 61672092.

References

1. B. AsSadhan, and J. M. Moura, "An Efficient Method to Detect Periodic Behavior in Botnet Traffic by Analyzing Control Plane Traffic," *Journal of advanced research*, vol. 5, no. 4, pp. 435-448, 2014
2. C. C. Aggarwal, Y. Zhao, and P. S. Yu, "Outlier Detection in Graph Streams," *International Conference on Data Engineering*, vol.6791, pp. 399-409, 2011
3. L. Akoglu, M. Mcglohon, and C. Faloutsos, "Anomaly Detection in Large Graphs," *Cmu-Cs-09-173-Technical Report*, 2009.
4. H. Choi, and H. Lee, "Identifying Botnets by Capturing Group Activities in DNS Traffic," *Elsevier North-Holland*, vol. 56, no. 1, pp. 20-33, 2012
5. G. Gu, R. Perdisci, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection," *Proceedings of the 17th Usenix Security Symposium*, pp.139-154, 2008

6. S. Gianvecchio, M. Xie, Z. Wu, and H. Wang, "Measurement and Classification of Humans and Bots in Internet Chat," *Usenix Security Symposium*, pp. 155-170, 2009
7. X. Guan, W. Wang, and X. Zhang, "Fast Intrusion Detection Based on A Non-negative Matrix Factorization Model," *J. Network and Computer Applications* 32 (1), pp. 31–44 2009
8. J. A. Hartigan, and M. A. Wong, "A K-means Clustering Algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100-108, 1979
9. G. Kirubavathi, and R. Anitha, "Botnets: A Study and Analysis," *Computational Intelligence, Cyber Security and Computational Models*, pp. 203-214, 2014
10. B. Perozzi, and L. Akoglu, "Focused Clustering and Outlier Detection in Large Attributed Graphs," *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1346-1355, 2014
11. W. Wang, T. Guyet, R. Quiniou, M. Cordier, F. Masseglia, and X. Zhang, "Autonomic Intrusion Detection: Adaptively Detecting Anomalies over Unlabeled Audit Data Streams in Computer Networks," *Knowledge-Based Systems*, vol.70 no. 11, pp. 103-117, 2014
12. W. Wang, X. Guan, and X. Zhang, "Processing of Massive Audit Data Streams for Real-time Anomaly Intrusion Detection," *Computer Communications* 31 (1), pp. 58–72, 2008
13. W. Wang, J. Liu, G. Pitsilis, and X. Zhang, "Abstracting Massive Data for Lightweight Intrusion Detection in Computer Networks," *Information Sciences*, vol. 433-434, no. 4, pp. 417-430, 2018
14. W. Wang, Y. Li, X. Wang, J. Liu, and X. Zhang, "Detecting Android Malicious Apps and Categorizing Benign Apps with Ensemble of Classifiers," *Future Generation Computer Systems*, vol.78, pp. 987–994, 2018
15. W. Wang, X. Wang, D. Feng, J. Liu, Z. Han, and X. Zhang, "Exploring Permission-induced Risk in Android Applications for Malicious Application Detection," *IEEE Transactions on Information Forensics and Security* 9 (11), pp. 1869–1882, 2014
16. X. Wang, W. Wang, Y. He, J. Liu, Z. Han, and X. Zhang, "Characterizing Android Apps' Behavior for Effective Detection of Malapps at Large Scale," *Future Generation Computer Systems*, 75: 30-45, 2017
17. A. Zimek, E. Schubert, and H. P. Kriegel, "A Survey on Unsupervised Outlier Detection in High-dimensional Numerical Data," *Statistical Analysis & Data Mining*, vol. 5, no. 5, pp. 363–387, 2012
18. D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, and A. Ghorbani, "Botnet Detection Based on Traffic Behavior Analysis and Flow Intervals," *Computers & Security*, vol. 39, no. 4, pp. 2-16, 2013
19. Z. Zhu, V. Yegneswaran, and Y. Chen, "Using Failure Information Analysis to Detect Enterprise Zombies," *Security and Privacy in Communication Networks*, vol. 19, pp. 185-206, 2009
20. "Spamhaus Botnet Threat Report 2017," Available at https://www.spamhaus.org/news/article/772/spamhaus-botnet-threat-report-2017, Last accessed on January 31, 2018
21. "What is Docker," Available at https://www.docker.com/what-docker, Last accessed on January 31, 2018

Wei Wang is a full professor in Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, School of Computer and Information Technology, Beijing Jiaotong University, China. He earned his PhD degree from Xi'an Jiaotong University in 2005. He was a postdoctoral researcher at the University of Trento, Italy, from 2005-2006 and at TELECOM Bretagne and at INRIA, France, from 2007-2008. He was an European ERCIM Fellow in the Norwegian University of Science and Technology (NTNU), Norway, and at the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, from 2009-2011. He visited INRIA, ETH, NTNU, CNR, and New York University Polytechnic. He was guest editor of IEEE Network and currently a young AE of Frontiers of Computer Science. He has authored or co-authored over 60 peer-reviewed papers in various journals and international conferences. His main research interests include network security and data mining.

Yang Wang is currently a M.S. student in the School of Computer and Information Technology, Beijing Jiaotong University, China. He received his B.S. degree from Shanxi University, China, in 2016. His main research interests lie in Botnet detection.

XinLu Tan is an undergraduate student in the School of Computer and Information Technology, Beijing Jiaotong University. Her major is Information Security.

Ya Liu is an undergraduate student in the School of Computer and Information Technology, Beijing Jiaotong University. Her major is Information Security.

Shuangmao Yang is a senior engineer in Science and Technology on Electronic Information Control Laboratory, Chengdu, China. He is a PhD holder. His research interests include wireless communication and signal and information processing techniques.