Microsyst Technol (2018) 24:209–217
https://doi.org/10.1007/s00542-016-3237-0

TECHNICAL PAPER

Bot detection using unsupervised machine learning

Wei Wu[1] · Jaime Alvarez[2] · Chengcheng Liu[3] · Hung-Min Sun[2]

Received: 30 October 2016 / Accepted: 7 December 2016 / Published online: 31 December 2016

Abstract This research focuses on bot detection through implementation of techniques such as traffic analysis, unsupervised machine learning, and similarity analysis between benign traffic data and bot traffic data. In this study, we tested and experimented with different clustering algorithms and recorded their accuracy with our prepared datasets. Later, the best clustering algorithm was used to proceed with the next steps of the methodology such as determination of majority clusters (cluster with most flows), removal of duplicate flows, and calculation of similarity analysis. Results were recorded for the removal of duplicate flows stage, the results indicate how many flows each majority cluster contains and how many duplicate flows were removed from this majority cluster. Next, results for similarity analysis indicate the value of the similarity coefficient for the comparisons between all datasets (bot datasets and benign dataset). With these results we can present some heuristic conclusion for determining possible bot infection in a certain host.

✉ Hung-Min Sun
hmsun@cs.nthu.edu.tw

Wei Wu
weiwu@fjnu.edu.cn

Chengcheng Liu
liuchengcheng600@126.com

[1] Fujian Provincial Key Laboratory of Network Security and Cryptology, School of Mathematics and Computer Science, Fujian Normal University, Fuzhou, Fujian, China

[2] Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan

[3] College of Computer and Control Engineering, Nankai University, Tianjin, China

1 Introduction

As with most popular consumer technologies, cloud storage are a potential and attractive target for criminals. Bot is a way. The term bot refers to a computer that has been compromised with sophisticated bot malware which puts them under the control of an attacker (Choo 2014; Debiao et al. 2012; Jiang and al 2014). Nowadays, botnets represent a serious threat in the Internet. For example, as noted by cyber criminologists that botnets can be used to facilitate a number of other cybercrimes (Choo 2007, 2008; Choo and Smith 2008; Choo and Grabosky 2014; Karim et al. 2016). A botnet consists of a group of bots that act in a coordinated manner to perform malicious activities in the Internet. These bots are pieces of malware that are used to compromise hosts in a network so that the host can be remote-controlled by the attacker, denominated botmaster (Osanaiye et al. 2016a, b).

Over the years, many different types of bots and botnets have been engineered, each one becoming more resilient, harmful, and intelligent, known from (Barford and Yegneswaran 2006). Botnet detection and destruction methods have also evolved, which employ different techniques such as honey pots, traffic analysis (Stevanovic and Pedersen 2014a; Brozycki 2010; Arndt 2016; Zhao et al. 2013) and machine learning (McGregor et al. 2004; Nivargi et al. 2016), just a few of them have been mentioned.

Botnet detection in this research focuses on traffic analysis under the basic premise that is flows generated by bots differ from normal benign flows in terms of behavior. With this premise, machine learning (clustering in this case) can be attempted to apply to group the flows into different clusters depending on their behavior with the possible highest accuracy, which is mentioned by (Zhao 2013). It is important to select the necessary instances and features by using

some methods, such as a scalable approach (Garca-Pedrajas et al. 2013), n-gram feature selection (Lu et al. 2011), and some analyzed features of traffic (Pohlmanna et al. 2013; Stevanovic and Pedersen 2013; Livadas 2006). Once the flows have been accurately separated into different clusters by using the appropriate algorithm, the bot flows can be analyzed and their similarity can be checked among other bot flow groups. Flows of the same bot [E.g. Waledac (Trolle Borup 2009)] will share a lot of similarity. Also, bot flows belonging to different bots (E.g. Waledac and Storm) will also share some similarity due to similar behavior.

Following this approach, it is also possible to do experiment with data captured from particular hosts and make deductions based on their similarity.

2 Related work

It seems that the predominant botnet detection methods like (Stevanovic and Pedersen 2014b) are supervised machine learning methods, mainly include classification, decision trees, Naive Bayes, and artificial neural networks like (Guntuku et al. 2013). The authors use semi-supervised ant colony clustering to detect botnet in (Huseynov et al. 2014). Some researches like (Su 2015; Saad et al. 2011; Hota et al. 2013) and (Singh et al. 2014; Narang et al. 2013; Huseynov and Kim 2014; Rahbarinia and Perdisci 2013) can detect p2p botnets, some like (Cai and Zou 2012; Alomari and Manickama 2014) can detect http botnets.

Some of the prior bot detection researches use unsupervised machine method mentioned in (Ghahramani 2004) by applying clustering technique through different methodologies. These works also differ in the features that are selected for machine learning.

2.1 BotOnus

BotOnus (Yahyazadeh and Abadi 2015) is an online unsupervised botnet detection method that does not require a priori knowledge of botnets. It extracts a set of flow feature vectors from the network traffic at the end of each period, and then groups them to some flow clusters by an on-line fixed-width clustering algorithm. Flow clusters that have at least two members, and whose intra-cluster similarity is above threshold, are identified as suspicious botnet clusters, and all hosts in such clusters are identified as bot infected.

BotOnus uses unsupervised machine learning detection and is also an on-line detection method, unlike other botnet detection systems. BotOnus is able to detect unknown botnets, since it uses an unsupervised technique driven by intrinsic characteristics of botnets such as group activities, without prior knowledge of them. The main characteristics of BotOnus is that just like this study and other clustering implemented frameworks (Osanaiye et al. 2016c, d; Peng et al. 2016), it implements a cluster removal criterion as well as a cluster similarity criterion.

2.2 Botminer

Botminer (Gu et al. 2008) is a framework for botnet detection using clustering based on data mining. In Botminer, it is assumed that bots within both centralized and P2P structures are likely to behave similarly and save similar communication patterns.

Botminer detects bots by performing cross-cluster correlations of similar communication patterns and malicious activities between its clusters. Separating the communication patterns in a C-Plane cluster, and malicious activity patterns in an A-Plane cluster.

2.3 Peershark

Peershark (Narang et al. 2014) is another clustering botnet detection framework. While Peershark implements clustering in its framework, it also uses supervised machine learning techniques such as decision trees, random forests, and bayesian networks. Hence, it can be considered a semi-supervised machine learning approach.

The main aim of Peershark is to detect botnets when they lie dormant or when they are performing malicious activities. Peershark works by extracting statistical features from network traces of P2P applications and botnets. The machine learning models are then capable of differentiating the traffic between the bots and the applications.

2.4 CluSiBotHealer

Our methodology was primarily inspired by CluSiBotHealer (Barthakur et al. 2015). However their approach differs in the clustering algorithm used (EM Clustering) as well as the selected features. The results of this paper included tests done by using EM clustering, and yielded extremely varied results. Simple-K Means was the chosen algorithm.

3 Materials and methods

Flows generated by bots differ from normal benign flows in terms of behavior. In order to utilize the difference and select the attributes for bot detection, this paper firstly prepared large amounts of real-word network data, and then do something for feature extraction to determine the attributes, finally, using the dataset to do clustering and similarity analysis. It is because the data is real, the conclusion

Table 1 ISOT dataset infected Storm hosts

IP address	Type of traffic	Label of malicious traffic
172.16.2.11	Malicious/UDP (Storm)	Src/Dst MAC BB:BB:BB:BB:BB:BB
172.16.0.12	Malicious/SMTP spam (Storm)	Src/Dst MAC AA:AA:AA:AA:AA:AA

can reflect whether the method in this paper can detect the latest botnets or not.

3.1 Datasets

Many datasets for botnet detection research and implementation exist. These datasets consist of large amounts of network data that contain normal traffic information as well as malicious and botnet traffic. Seperate datasets containing benign P2P traffic captures as well as Zeus, Waledac, and Storm traffic captures were used for machine learning.

The bot traces for the datasets Zeus and Waledac were obtained from the University of Georgia, USA. For Storm traces, partitions from the ISOT dataset were made in order to extract the flows corresponding with known Storm infected hosts.

Two datasets were prepared for each bot, with a mix of both bot flows and benign flows. It is worth noting that the flows of botnet traffic captures are not 100% malicious. This is because that the infected host might make DNS queries. However, the flows not resolved through DNS that are between the infected hosts and other IP addresses are indeed malicious. So during our flow labeling step, we must take the above information into consideration when labeling flows from these infected hosts. Additionally, one more dataset consisting of only benign flows was prepared in order to compare it with the other datasets. Finally, all the flows were labeled as malicious or non-malicious in order to obtain clustering error values as well as feature rank values (next section), Table 1 shows sample information of the dataset.

3.2 Feature extraction and selection

Firstly, Wireshark, a network protocol analyzer need to be used to obtain a .pcap file with sorted packet tracing data, which is known as traffic capturing. Network traffic must be first analyzed in a flow level manner. In most of botnet detection research, packets are classified as flows or conversations based on the 5-tuple: Src. IP, Dest. IP, Src. port, Dest. Port, Proto. After there being a significant amount of packet traffic data, it is need to extract the flows from this data along with some features. Statistical flow features were extracted using a tool called Netmate, which has also been used in other botnet detection work in previous work. Approximately 45 features were extracted.

Then, carry out feature selection from 45 features, with the purpose of obtaining better results and having a better understanding of which features (or attributes) are the most relevant when applying machine learning algorithms of this paper.

The process of feature selection was executed by applying Information Gain, which basically evaluates each feature and its influence on the data according to the class (the label). This whole process results in relevant features that consequently guarantee high accuracy when performing clustering later.

The selected features are:

(1) dstport: The destination port number.
(2) max bpktl: The size of the largest packet sent in the backward direction (in bytes).
(3) max fptkl: The size of the largest packet sent in the forward direction (in bytes).
(4) fpsh cnt: The number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP).
(5) min fptkl: The size of the smallest packet sent in the forward direction (in bytes).

3.3 Clustering

All the datasets were clustered using the "Weka machine learning suite" by applying Simple-K means clustering algorithm, which yielded the most accurate results. Flows were clustered into two clusters which represent malicious flows and non-malicious flows. The clustering results were saved into an output file in .arff format.

It is very likely that the bot clusters will contain duplicated flows. Duplicated flows are defined as flows that share the same values for the selected features. After determining the majority cluster, Java application should be able to immediately proceed to analyze pairs of flows within the bot cluster, and can check their feature values to determine if they are duplicated or not. Duplicated flows will be removed from the cluster.

The removal of duplicated flows will help make the computation of the Jaccard Coefficient be much easier and faster for the application. However, a large reduction of duplicate flows within the bot cluster can also indicate that the bot is a P2P bot, in fact. This is because a large number of P2P bot flows share the same packet and flow structure due to repeated the same commands transmission through different ports.

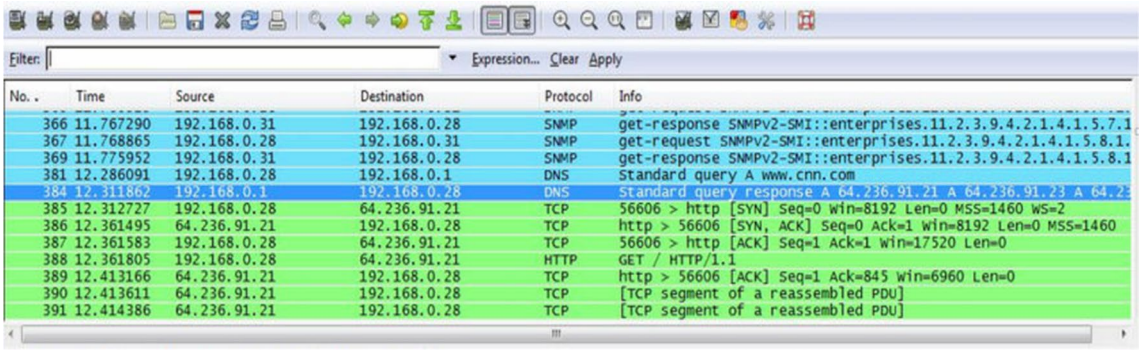

Fig. 1 Wireshark packet tracing

3.4 Similarity analysis

It can be proceeded to separate the flows only belonging to the malicious cluster by using the cluster output file as input. Then, these flows can be compared to other malicious flows for their similarity, using Jaccard Similarity defined as (1):

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, \quad 0 \leq J(A,B) \leq 1. \tag{1}$$

Clusters that are completely different from each other will yield a value of 0.0.

4 Results and discussion

From the above, what have already been known is that raw data gained from different ways should be cleaned, filtered and pre-processed by feature extraction, data labeling, and feature selection, which need to be converted into datasets in specific format for Weka (Fowler and Robert 2014) with a collection of machine learning algorithms. Accuracy outcomes of Simple K-Means, X-Means, and EM Clustering algorithms will be presented in this section.

Before actually proceeding to extract flow features, we must first perform sort of packet tracing or obtain packet tracing data. This process is also known as traffic capturing and consists on capturing the traffic of specific machines that belong to a certain network. Packet tracing data is stored in files with .pcap extension. In order to obtain a .pcap file with packet tracing data we need to use a tool called Wireshark. Wireshark is a network protocol analyzer that allows us to make packet captures in a specified network. Wireshark will trace all the packets that travel through the network until the capture ends.

The example packet data shown in Fig. 1 can be exported as a .pcap file so that we can later start processing it for flow statistics extraction with other available tools.

For dataset preparations, six datasets were prepared in total, two datasets for each bot. Each dataset contains a random selection of 15,000 flows from its respective bot, with additional 5000 random benign flows added, making a total of 20,000 combined flows for each dataset.

It is worth mentioning that among the 15,000 bot flows, there is a very small amount of benign flows (due to DNS querying, as previously explained) present. However, the amount of these flows is very minimal and does not alter the benign-malicious ratio in a significant way.

4.1 Data pre-processing: data labeling

Because this paper focuses on unsupervised machine learning, we need to work with unlabeled data. This means that the data inside the dataset does not contain labels. For example, in a botnet detection dataset, each flow could already be pre-labeled as malicious or non-malicious. However, in order to have an idea or insurance that our clustering algorithms are correctly clustering most of the flows in the data, we first want to work with labeled data. This data labeling must be done manually by us. Since we are dealing with relatively big amounts of data and flows, we created a small data labeling module in our Java application. This module basically takes the .ARFF file and reads each flow line by line after the @data statement. We take the srcip field and compare it to the list of IPs of infected hosts mentioned previously. Since we have separate clean and botnet traffic capture files for each bot, we know that the majority of the flows in these botnet traffic captures are indeed malicious.

However, as mentioned before, this does not mean that 100% of the flows are malicious (DNS queries). This is why the labeling module needs to analyze each flow and label it carefully. Therefore, if the flow uses protocol UDP with known DNS ports 53 or 5353, or if the flow uses protocols that are not UDP or TCP, we label that flow as non-malicious.

The above procedure is repeated for all the traces for each botnet. For the clean P2P traces, all those are labeled

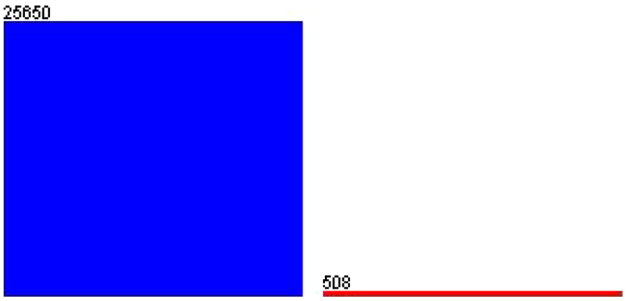

Fig. 2 Visualization of labeled Zeus flows. *Blue flows* indicate malicious flows, while *red flows* indicate non-malicious flows

Table 2 Zeus attribute evaluation results

#	Name	Rank (dataset 1)	Rank (dataset 2)
1	dstport	0.686	0.6769
2	max_bpktl	0.5854	0.5907
3	max_fptkl	0.704	0.6935
4	fpsh_cnt	0.0795	0.074
5	min_fpktl	0.6922	0.6944

as non-malicious by default. This will leave us with two labels in our datasets: malicious, and non-malicious. For this to work with the ARFF relation, it is need to create a new Class field so that Weka can recognize these labels as classes:

This way Weka will be able to read and create this new attribute. We can then visualize the labeled data in Weka according to our labeling, as shown in Fig. 2. The nonmalicious flows in Fig. 2 (shown in red) represent the DNS queries found in the Zeus flows, detected by our Labeling module.

4.2 Feature selection & attribute evaluation results

Careful selection of features for the datasets was needed in order to obtain the minimal amount of clustering error. However, attribute evaluation for different bot datasets yielded different features. Since the exactly same features were needed in all different bot datasets in order to make the Jaccard similarity coefficient work and be accurate, a careful selection of features was made based on the minimum percentage of incorrectly clustered instances in all bot datasets.

The following are the infected host's IP addresses for the botnet traces:

1. Zeus: 10.0.2.15.
2. Waledac: 192.168.58.136, 192.168.58.137, 192.168.58.150.

Table 3 Waledac attribute evaluation results

#	Name	Rank (dataset 1)	Rank (dataset 2)
1	dstport	0.841	0.8396
2	max_bpktl	0.766	0.7653
3	max_fptkl	0.739	0.7347
4	fpsh_cnt	0.711	0.7023
5	min_fpktl	0.702	0.6944

Table 4 Storm attribute evaluation results

#	Name	Rank (dataset 1)	Rank (dataset 2)
1	dstport	0.6861	0.6483
2	max_bpktl	0.6455	0.5175
3	max_fptkl	0.8457	0.765
4	fpsh_cnt	0.1919	0.0469
5	min_fpktl	0.7742	0.8043

The following tables (i.e. Tables 2, 3, 4) indicate the selected features for all datasets, with their respective rank and percentage of incorrectly clustered instances. The features displayed in the tables below are the features that were selected for all botnet datasets because they yielded a minimal and acceptable error margin when using clustering for each botnet dataset.

What can be seen clearly are that there are some drastic rank changes in some features depending on the bot. For example, in Zeus as shown in Table 2, fpsh_cnt's rank is incredibly low compared to Waledac's (Tables 3, 4).

4.3 Clustering performance results

It can be proceeded to separate the flows only. Different clustering algorithms were used to record experiments and discover which clustering algorithm was more efficient. The outcome of Tables 5, 6, 7 for the experiments showed that Simple-K Means algorithm was the most efficient on average between all the datasets.

The results for all datasets showed by X-Means is similar as that by Simple-K Means. Performance of EM (Expectation Maximization) algorithm varied greatly depending on the dataset and features. The way to determine the efficiency of each clustering algorithm is by using the percentage of incorrectly clustered instances. This value is available when using labeled flows to determine how many instances are incorrectly clustered by the algorithm:

Clustering accuracy performance using Simple-K Means was measured by using (2):

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \quad (2)$$

Table 5 Incorrectly clustered instances for ZEUS datasets

Algorithm	Dataset 1 (%)	Dataset 2 (%)
Simple-K Means	4.85	4.98
X-Means	4.85	4.98
EM	27.34	27.285

Table 6 Incorrectly clustered instances for WALEDAC datasets

Algorithm	Dataset 1 (%)	Dataset 2 (%)
Simple-K Means	4.82	4.975
X-Means	4.82	4.77
EM	0.15	1.035

Table 7 Incorrectly clustered instances for STORM datasets

Algorithm	Dataset 1 (%)	Dataset 2 (%)
Simple-K Means	11.42	2.46
X-Means	11.42	2.46
EM	28.85	8.19

Table 8 Performance metrics

	TP	TN	FP	FN	Accuracy
Zeus 1	14,678	4352	969	1	0.9515
Zeus 2	14,663	4341	991	5	0.9502
Waledac 1	14,536	4500	963	1	0.9518
Waledac 2	14,521	4525	963	1	0.9523
Storm 1	10,139	4499	501	1386	0.8858
Storm 2	2300	503	247	3	0.9181

with the above formula and the above results for incorrectly clustering instances, the number of true positives, true negatives, false positives and false negatives can all be obtained to calculate the accuracy value, and the accuracy values are shown as Table 8:

4.4 Flow clustering

Once we have correctly pre-processed our data, we can begin by working with Weka's clustering algorithms and our botnet datasets. Weka offers many clustering algorithms, but we will focus on Simple K-Means, X-Means, and EM clustering algorithms. Because what we want are clustering benign and malicious traffic flows. We want to cluster our datasets in two clusters, for all the clustering algorithms.

As previously mentioned, we will firstly work with labeled data in order to analyze the clustering behavior and accuracy. Classes to Clusters is a clustering mode in Weka that can help us achieve this. This mode basically clusters the data with the selected features normally, ignoring the label/class attribute. Later it will compare the clustered flows with the labels in the data and will compute a value of incorrectly clustered instances. This tells us how well the clustering is working, and it can be an indicator of the efficiency of a particular clustering algorithm, or say, it can tell us whether the different features should be used.

As shown in Fig. 3, we can see the amount of instances found in each cluster after the clustering. We can also see how many instances were placed in the wrong cluster.

We can also visualize these incorrectly clustered flows/instances by using Weka's cluster assignments visualization feature. As shown in Fig. 4, each color represents a cluster, and the squares indicate the incorrectly clustered flows by using Classes to Clusters evaluation with different feature combinations, we can aim to obtain a minimal or

```
Class attribute: class
Classes to Clusters:

     0     1  <-- assigned to cluster
  9911 22959 | Malicious
   583  1287 | Non-Malicious
    40   295 | Unknown

Cluster 0 <-- Non-Malicious
Cluster 1 <-- Malicious

Incorrectly clustered instances :       11533.0  32.881  %
```

Fig. 3 Example output of a classes to clusters evaluation using simple K-means

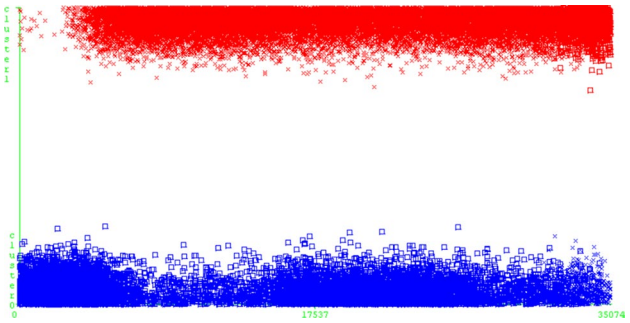

Fig. 4 Example of classes to clusters cluster visualization in Weka

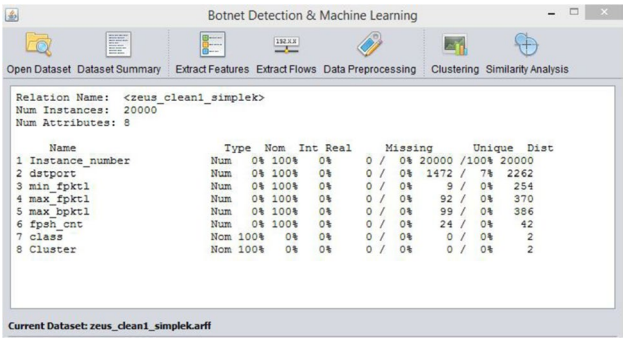

Fig. 5 Screenshot of our Java application

satisfactory value of incorrectly clustered instances. When we have obtained this, we can proceed on working with unlabeled data.

After a dataset has been clustered, we can proceed to save this clustered dataset if we consider that the percentage of incorrectly clustered instances is satisfactory. When saving the clustered dataset, Weka automatically appends a new cluster attribute similar to our label attribute. The difference is that this new attribute represents the cluster of the flow assigned by the clusterer:

@attribute Cluster {cluster0,cluster1}

4.5 Determining majority clusters

Since our prepared datasets contain more botnet flows than benign flows. If our clustering accuracy is high, it is very likely that the cluster containing the malicious flows will have most of the flows from the dataset.

Also, when capturing traffic from an infected host. The flows from these captures will mostly be botnet flows. Therefore, we want to pay special attention to the cluster containing the majority of the flows (Fig. 5).

In our Java application like Fig. 5, before computing the Jaccard Similarity coefficient, the application determines which cluster is the majority cluster from each dataset by comparing the amount of flows between the malicious and non-malicious clusters from each dataset.

4.6 Duplicate flow reduction

Duplicate flows were removed from each of the majority (or suspected as bot) clusters in each dataset. Table 9 describes the results obtained after performing the flow reduction.

It is important to remember that these clusters are possible bot clusters, according to the clustering algorithm. Hence, most (given the low clustering error value) of its flows will be indeed bot flows.

A significant reduction of duplicate flows can indicate that the host might be infected with a possible P2P bot. This is because P2P bot flows share the same packet and flow structure because of repeated transmission of the same commands through different ports.

What can be noticed is that Benign 1 had a very significant flow reduction. This is to be expected, since Benign 1 dataset is composed of flows obtained from P2P braces, which share P2P flow behaviors.

4.7 Similarity analysis

Table 10 shows the similarity analysis results obtained between all datasets. Note the comparison of similarity between different bots, as well as the comparison between bots and benign clusters (Table 10).

Since two datasets are prepared for each bot. We will proceed to calculate the Jaccard Similarity Coefficient between the bot cluster of each dataset, for each bot. In

Table 9 Results of removal of duplicated flows in majority clusters

Dataset	Cluster flows	Removed flows	Percentage
Zeus 1	15,647	5090	33
Zeus 2	15,654	5041	32
Waledac 1	15,499	9444	61
Waledac 2	15,474	9361	60.50
Storm 1	10,640	4976	46.77
Storm 2	2376	290	12.21
Benign 1	12,675	9674	76.32

Table 10 Similarity analysis results

	Zeus 2	Waledac 2	Storm 2
Zeus 1	0.0766	0.0231	0.0036
Waledac 1	0.0328	0.4242	0.0043
Storm 1	0.0106	0.0215	0.2086
Benign	0.00	0.00	0.00

other words, the calculation is done by dividing the size of the intersection between the size of the union, of the both sets (bot clusters). The result will be a coefficient that will indicate more or less how similar both bot clusters are. This will help to determine if the host is indeed bot-infected or not.

Botnet detection using traffic analysis, machine learning and similarity analysis has some flaws, as seen with the similarity result between Zeus datasets. Possible improvements for future work could be finding ways to bypass encrypted traffic in order to obtain more robust results in the similarity analysis. Another idea would be working with different features from those Netmate provides, and attempt to reduce the error value even lower than those from the results conducted in the experiments. Other existing clustering algorithms not covered in this reserach could also be used to achieve this.

5 Conclusion

This research implemented and tested botnet detection techniques using unsupervised machine learning through various clustering algorithms. As noted by a number of researchers (Quick and Choo 2016a, b; Quick and Choo 2014a, b; Quick and Choo 2013a, b, c; Do et al. 2016; Martini and Choo 2012, 2013, 2014; Ab Rahman et al. 2016; Cahyani et al. 2016), using real-world datasets to evaluate a mitigation or detection strategy helps to demonstrate the usefulness of the technique such as ours in this paper. The data was preprocessed, clustered and further analyzed through some similarity metrics in order to identify possible similarities between benign data flows and bot infected data.

From the results, what can be clearly seen is that all bot flow data share some similarity among the different types of bots. However, bot flow data and benign flow data share no similarity whatsoever (0.0 result). This leads us to believe that a comparison with flow data from an unknown botnet would yield some similarity as well, and this could serve as a starting point for further inspection.

Based on these results, some heuristics that can help us come up with a final answer regarding an infected host can be defined. This paper concludes that if a dataset had a similarity coefficient greater or equal than 0.1 when compared to any of these bot datasets, then the host will be possibly infected with a different bot. If the similarity was greater or equal than 0.4, then the host will be very likely to be infected with the bot that is being compared to.

Regardless of the similarity results, what must be kept in mind is that this method has its flaws. For example, when the flow data is encrypted (as in the case of results obtained with Zeus), the algorithms results will be affected, so, the data must be carefully preprocessed in order to produce the accurate results.

Acknowledgements The authors would like to thank Pratik Narang and Babak Rahbarinia for their immense help and insights. And, we sincerely thank the anonymous reviewers and our shepherd for their constructive comments and valuable feedback which helped to improve this paper.

References

Ab Rahman NH, Cahyani NDW, Choo KKR (2016) Cloud incident handling and forensic-by-design: cloud storage as a case study. Concurrency and Computation: Practice and Experience

Cahyani NDW et al (2016) Forensic data acquisition from cloud-of-things devices: windows Smartphones as a case study. Concurrency and Computation: Practice and Experience

Alomari E, Manickama S (2014) Design, deployment and use of http-based botnet testbed. National Advanced IPv6 Centre (NAv6), Universiti Sains Malaysia, Malaysia. 16th International Conference on Advanced Communication Technology, 1265–1269 **(IEEE)**

Arndt D (2016) How to: calculating flow statistics using netmate. https://dan.arndt.ca/nims/calculating-flowstatistics-using-netmate/. Accessed 04 Dec 2016

Barford P, Yegneswaran V (2006) An inside look at botnets. In Special Workshop on Malware Detection, Advances in Information Security, Springer Verlag

Barthakur P, Dahal M, Ghose MK (2015) Clusibothealer: botnet detection through similarity analysis of clusters. J Adv Comp Netw 3:1

Brozycki J (2010) Capturing and analyzing packets with perl. SANS Institute InfoSec Reading Room

Cai T, Zou F (2012) Detecting http botnet with clustering network traffic. In: Wireless Communications, Networking and Mobile Computing (WiCOM), 8th International Conference on IEEE

Choo KR (2007) Zombies And botnets. trends and issues in crime and criminal justice. Australian Institute of Criminology Canberra, Justice 333:1–6

Choo KK (2008) Raymond organised crime groups in cyberspace: a typology. Trends organ crime 11(3):270–295

Choo K-KR (2014) Mobile Cloud Storage Users. IEEE Cloud Comput 1(3):20–23

Choo KKR, Grabosky P (2014) Cyber crime. In: Paoli L (ed) Oxford handbook of organized crime. Oxford University Press, New York, pp 482–499

Choo KKR, Smith RG (2008) Criminal exploitation of online systems by organised crime groups. Asian J criminol 3(1):37–59

Debiao H, Jianhua C, Rui Z (2012) A more secure authentication scheme for telecare medicine information systems. J Med Syst 36(3):1989–1995

Do Q, Martini B, Choo KKR (2016) Is the data on your wearable device secure? An Android Wear smartwatch case study. Software: Practice and Experience

Fowler CA, Robert JH (2014) Converting PCAPs into Weka mineable data. Department of Computer and Information Sciences Towson University. Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 15th IEEE/ACIS International Conference on IEEE

Garca-Pedrajas N, de Haro-Garca A, Prez-Rodriguez J (2013) A scalable approach to simultaneous evolutionary instance and feature selection. Inf Sci 228:150–174

Ghahramani Z (2004) Unsupervised learning. Gatsby Computational Neuroscience Unit University College London, UK. In: Advanced lectures on machine learning. Springer, Berlin, Heidelberg, pp 72–112

Gu G, Perdisci R, Zhang J, Lee W (2008) Botminer: clustering analysis of network traffic for protocol- and structure-independent botnet detection. College of Computing, Georgia Institute of Technology. USENIX Security Symposium, vol. 5, no. 2

Guntuku SC, Narang P, Hota C (2013) Real-time peer-to-peer botnet detection framework based on bayesian regularized neural network. Institute of Network Engineering College of Computer Science National Chiao Tung University. arXiv preprint arXiv:1307.7464

Hota C, Narang P, Reddy JM (2013) Feature selection for detection of peer-to-peer botnet traffic. In: Proceedings of the 6th ACM India Computing Convention

Huseynov K, Kim K (2014) Unsupervised hadoop-based p2p botnet detection with threshold setting. Department of Computer Science, Korea Advanced, Institute of Science and Technology

Huseynov K, Kim K, Yoo PD (2014) Semi-supervised botnet detection using ant colony clustering. SCIS 2014. In: The 31th symposium on cryptography and information security Kagoshima. The Institute of Electronics, Information and Communication Engineers, Japan

Jiang T, Chen X et al (2014) TIMER: secure and reliable cloud storage against data re-outsourcing. In: International Conference on Information Security Practice and Experience. Springer International Publishing

Karim Ahmad et al (2016) On the analysis and detection of mobile botnet applications. J Univ Comput Sci 22(4):567–588

Livadas C (2006) Using machine learning techniques to identify botnet traffic. Internetwork Research Department BBN Technologies. In: Proceedings 31st IEEE Conference on Local Computer Networks, p 967–974

Lu W, Rammidi G, Ghorbani AA (2011) Clustering botnet communication traffic based on n-gram feature selection. Comp Commun 34(3):502–514

Martini B, Choo KKR (2012) An integrated conceptual digital forensic framework for cloud computing. Digit Invest 9(2):71–80

Martini B, Choo KKR (2013) Cloud storage forensics: ownCloud as a case study. Digit Invest 10(4):287–299

Martini B, Choo KKR (2014) Distributed filesystem forensics: XtreemFS as a case study. Digit Invest 11(4):295–313

McGregor A, Hall M, Lorier P, Brunskill J (2004) Flow clustering using machine learning techniques. The University of Waikato, New Zealand

Narang P, Reddy JM, Hota C (2013) Feature selection for detection of peer-to-peer botnet traffic. Department of Computer Science & Engineering Birla Institute of Technology and Science-Pilani. In: Proceedings of the 6th ACM India Computing Convention, p 16

Narang P, Hota C, Venkatakrishnan VN (2014) Peershark: flow-clustering and conversation generation for malicious peer-to-peer traffic identification. EURASIP J Inf Sec

Nivargi V, Bhaowal M, Lee T (2016) Machine learning based botnet detection. Citeseer. http://www.stanford.edu/class/cs229/proj2006/NivargiBhaowalLeeMachineLearningBasedBotnetDetection.pdf. Accessed 10 Oct 2006

Osanaiye O et al (2016a) Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. EURASIP J Wirel Commun Netw 2016(1):1

Osanaiye O, Choo KKR, Dlodlo M (2016b) Analysing feature selection and classification techniques for DDoS detection in cloud. In: Proceedings of Southern Africa Telecommunication

Osanaiye O, Choo KKR, Dlodlo M (2016c) Distributed denial of service (DDoS) resilience in cloud: review and conceptual-cloud DDoS mitigation framework. In: Journal of network and computer applications, Networks and Applications Conference, 3–7, SATNAC, vol 67, pp 147–165

Osanaiye O, Choo KKR, Dlodlo M (2016d) Change-point cloud DDoS detection using packet inter-arrival time. In: Proceedings of IEEE Computer Science & Electronic Engineering Conference, 28–30, IEEE CEEC

Peng J, Choo KKR, Ashman H (2016) User profiling in intrusion detection: a review. J Netw Comp Appl 72:14–27

Pohlmanna N, Dietricha CJ, Rossowa C (2013) Cocospot: clustering and recognizing botnet command and control channels using traffic analysis. Comp Netw 57(2):475–486

Quick D, Choo KKR (2013a) Digital droplets: Microsoft SkyDrive forensic data remnants. Futur Gener Comp Syst 29(6):1378–1394

Quick D, Choo KKR (2013b) Dropbox analysis: data remnants on user machines. Digit Invest 10(1):3–18

Quick D, Choo KKR (2013c) Forensic collection of cloud storage data: does the act of collection result in changes to the data or its metadata? Digit Invest 10(3):266–277

Quick D, Choo KKR (2014a) Data reduction and data mining framework for digital forensic evidence: storage, intelligence, review, and archive. Trends Issues Crime Crimin Justice 480:1–11

Quick D, Choo KKR (2014b) Google drive: forensic analysis of data remnants. J Netw Comp Appl 40:179–193

Quick D, Choo KKR (2016) Big forensic data reduction: digital forensic images and electronic evidence. Clust Comput 19(2):723–740. doi:10.1007/s10586-016-0553-1

Quick D, Choo KKR (2016) Big forensic data management in heterogeneous distributed systems: quick analysis of multimedia forensic data. Software: Practice and Experience

Rahbarinia B, Perdisci R (2013) Peerrush: Mining for unwanted p2p traffic. Dept. of Computer Science, University of Georgia. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment pp. 62-82. Springer Berlin Heidelberg

Saad S, Traore I, Ghorbani AA, Sayed B, Zhao D, Lu W, Felix J, Hakimian P (2011) Detecting p2p botnets through network behavior analysis and machine learning. In: Proceedings of 9th Annual Conference on Privacy, Security and Trust (PST2011)

Singh K, Guntuku SC, Thakur A, Hota C (2014) Big data analytics framework for peer-to-peer botnet detection. Network 3:0 **(Elsevier)**

Stevanovic M, Pedersen JM (2013) Machine learning for identifying botnet network traffic. Department of Computer Science & Engineering Birla Institute of Technology and Science-Pilani

Stevanovic M, Pedersen M (2014) An efficientflow-based botnet detection using supervised machine learning. In: International Conference on Computing, Networking and Communications (ICNC), Honolulu, p 797–801

Stevanovic M, Pedersen JM (2014) An efficient flow-based botnet detection using supervised machine learning. Department of Electronic Systems, Aalborg University. In Computing, Networking and Communications (ICNC), 2014 International Conference on p797–801 **(IEEE)**

Su SC (2015) Detecting p2p botnet in software defined network. Institute of Network Engineering College of Computer Science National Chiao Tung University. http://ndltd.ncl.edu.tw/cgi-bin/gs32/gsweb.cgi/login?o=dnclcdr&s=id=%22103NCTU5726023%22.&searchmode=basic

Trolle Borup L (2009) Peer-to-peer botnets: a case study on Waledac. Dissertation, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark

Yahyazadeh M, Abadi M (2015) Botonus: an online unsupervised method for botnet detection. ISC Int J Inf Sec 4:1

Zhao D (2013) Botnet detection based on traffic behavior analysis and flow intervals. Comp Secur 39:2–16

Zhao D, Traore I, Sayed B (2013) Botnet detection based on traffic behavior analysis and flow intervals. Elsevier, Amsterdam