Unsupervised detection of botnet activities using frequent pattern tree mining

Hao, Siqiang; Liu, Di; Baldi, Simone; Yu, Wenwu

Published in:
Complex and Intelligent Systems

DOI:
10.1007/s40747-021-00281-5

Complex & Intelligent Systems (2022) 8:761–769
https://doi.org/10.1007/s40747-021-00281-5

ORIGINAL ARTICLE

Unsupervised detection of botnet activities using frequent pattern tree mining

Siqiang Hao[1] · Di Liu[1] · Simone Baldi[1,2] · Wenwu Yu[1,2]

Received: 7 September 2020 / Accepted: 19 January 2021 / Published online: 25 February 2021

Abstract

A botnet is a network of remotely-controlled infected computers that can send spam, spread viruses, or stage denial-of-service attacks, without the consent of the computer owners. Since the beginning of the 21st century, botnet activities have steadily increased, becoming one of the major concerns for Internet security. In fact, botnet activities are becoming more and more difficult to be detected, because they make use of Peer-to-Peer protocols (eMule, Torrent, Frostwire, Vuze, Skype and many others). To improve the detectability of botnet activities, this paper introduces the idea of association analysis in the field of data mining, and proposes a system to detect botnets based on the FP-growth (Frequent Pattern Tree) frequent item mining algorithm. The detection system is composed of three parts: packet collection processing, rule mining, and statistical analysis of rules. Its characteristic feature is the rule-based classification of different botnet behaviors in a fast and unsupervised fashion. The effectiveness of the approach is validated in a scenario with 11 Peer-to-Peer host PCs, 42063 Non-Peer-to-Peer host PCs, and 17 host PCs with three different botnet activities (Storm, Waledac and Zeus). The recognition accuracy of the proposed architecture is shown to be above 94%. The proposed method is shown to improve the results reported in literature.

Keywords Botnet detection · Internet security · Frequent pattern tree · Data mining

Introduction

With the continuous development of the Internet, the network has expanded from an interconnection of PCs to a mobile Internet. With the advent of 5G technology, further expansion is expected towards the Internet of Things and the Internet of Everything scenarios [1]. As a result, the amount of information exchanged over the Internet has reached unprecedented levels, but so have the threats and the need for security.

Today, botnets have become one of the major threats to Internet security [25]. A botnet attack typically occurs as follows: attackers invade a large number of hosts and implant virus programs through various software equipment vulnerabilities, sending phishing emails, or brute force cracking. Any host that is infected becomes a member of the botnet, and falls under the control of the bot virus program. A botnet can eventually stage several malicious activities, such as sending spam, stealing private information, or launching denial-of-service (DoS) attacks. Because the infected host is typically referred to as a 'zombie host', another popular name for a botnet is 'zombie network'.

The first botnet program was SubSeven 2.1 and was created in June 1999 (the interested reader can check https://f-secure.com/v-descs/subseven.shtml). SubSeven 2.1 relied on the IRC (Internet Relay Chat) protocol to control a large number of zombie hosts. Since 1999, a large number of bot virus programs based on the IRC protocol have appeared, and examples include GTBot, Sdbot, etc. [4,6]. Fortunately, IRC-based botnets can be easily defeated by shutting down and restarting the IRC server. However, attackers have found new protocols through which delivering the botnet activities, so that botnets based on HTTP protocol and P2P (Peer-to-Peer) protocol have appeared [24,32]. The latter have the most dangerous characteristics in terms of decentralization and

✉ Wenwu Yu
wwyu@seu.edu.cn

Siqiang Hao
haosiqiang@163.com

Di Liu
liud923@126.com

Simone Baldi
s.baldi@tudelft.nl

1 School of Cyber Science and Engineering, Southeast University, Nanjing, China

2 School of Mathematics, Southeast University, Nanjing, China

strong resilience, since P2P-based botnets cannot be easily shut down like IRC-based botnets and its activities are more difficult to detect [26,27]. The emergence of more and more P2P-based botnet programs poses a great threat to Internet security [5,20,30]. Dangerous botnet activities are also more and more observed in Internet-of-Things (IoT) environments [9,29] and social Internet-of-Things (IoT) environments [28].

Popular detection methods for P2P-based botnets are signature-based intrusion detection systems [8], which are similar to antivirus software and firewalls. However, packet encryption will invalidate such methods. The authors in [21] proposed a detection model, named detection by mining regional periodicity (DMRP), based on capturing the event time series, mining the hidden periodicity of host behaviors, and evaluating the mined periodic patterns to identify P2P bot traffic. The authors of [18] proposed a three-layer filtering botnet detection system, which is responsible for packet filtering, P2P application packet filtering, and P2P botnet detection, respectively. High accuracy with low false alarm rate have been reported by using such periodicity based methods, although the behavior characteristics considered for the botnet is too simplistic as compared to real-life botnets.

In order to handle the detection of more complex botnet activities, methods based on machine learning [2,10–12, 14,16,17,23] have been commonly used. A brief account for these methods is given hereafter: [16,17] combined a stream-based method and a session-based method to design a two-layer structure machine learning classifier that can distinguish between zombie P2P activities and normal P2P activities. Note that [16] improves the work of [17]. The method in [10] clustered the normal and abnormal P2P behavior; [23] compared, based on existing datasets, five commonly-used machine learning techniques for online botnet detection. The results of the evaluation show that it is possible to detect effectively botnets during the botnet Command-and-Control (C&C) phase and before bots launch their attacks using traffic behaviors only. Based on the characteristics of unlimited network data flow and drift concept, [12] proposed a multi-layer multi-classifier group detection system based on the research of single classifier and multi-classifier to store the optimal K-classifiers. All machine learning algorithms are faced with similar problems, most notably, the long training time [11,14]. Another crucial problem is the need for labeled examples, i.e., each classifier must be trained for a specific type of botnet, which makes the classifier in general unable to handle new/unknown botnet activities for which labelled examples are unavailable [29].

In view of this observation, methods in [7,13,31,32] are all based on botnet behavior, i.e. they classified botnet behavior based on time intervals without having seen a complete network flow. There are, however, several challenges which must be overcome to realize a full implementation of such behaviour-based detection systems, such as the lack of scalability to huge datasets, and the need for installing individual detectors on every network device and on any networks with more than a few hundred nodes.

The advantages and flaws of the different methods adopted in the literature, motivate us towards pursuing a different approach to the detection of botnet activities. In this paper, we exploit and tailor the Frequent Pattern Tree to botnet detection. Frequent Pattern Tree is a data mining method used for frequent pattern mining (also known as Association Rule Mining). The purpose of the algorithm is to discover frequent patterns or associations from data sets. Because the method can automatically detect rules, it does not need to be trained from specifically labeled botnet activities as in machine learning approaches. In addition, because the data set is stored in a tree structure called Frequent Pattern Tree, frequent items can be found by simply traversing the data set twice. The tree structure results in higher efficiency and lower runtime cost as compared to the other data mining algorithms. For example, it was shown that when the number of records in the data set is relatively large, the Frequent Pattern Tree algorithm has a significant advantage over the Apriori algorithm in terms of speed [15]. Speed and memory efficiency also makes the Frequent Pattern Tree algorithm widely used in search engines [3].

Most botnet detection approaches rely on machine learning algorithms, which have a long training time cost and are targeted to deal with known (labeled) botnet types. In the presence on unknown botnet types and large amount of data, the performance of machine learning methods might deteriorate seriously. The contribution of this article is the first introduction of the frequent item mining algorithm Frequent Pattern Tree in the field of botnet detection. The proposed approach relies on discovering essential characteristics of scriptability and frequent similarity in the underlying communication of P2P botnets: therefore, it can cope with different types of P2P botnets without the need for the different types to be labeled. In fact, it is shown via the PeerRush Dataset [22] that botnet activities can be detected and classified automatically, without presetting or pre-training for specific botnet types. Also, the proposed approach can process tens of millions of data sets in around half a minute, which is again shown via the PeerRush Dataset [22] with tens of millions of data. Extensive experiments show that, as compared with machine learning methods reported in literature for the same data set, the proposed methods improves in terms of efficiency and accuracy.

The remainder of this paper is organized as follows. The Frequent Pattern Tree data mining algorithm is recalled in second section. In third section, we introduce the implementation steps of the proposed detection system. Experimental results are illustrated in fourth section. Followed by conclusions in last section and evaluation of ideas for future work.

Algorithm 1 Frequent Pattern Tree: blacknote that `an` refers to the n-th element in `A`, `rm` refers to the m-th element in `R`, `listg` refers to the g-th element in `List`

```
Input: Record set R with attributes belongs to attribute set A
Output: Rule set containing frequent item pair set
{a1,a2,a3,..,an}←Attribute set A
{r1,r2,blackr3,..,rm}←Record set R
Cai←Count of Attribute ai
S←Minimum Support
Bai←Conditional Pattern Base of ai
C←Item Header List i
List={list1,list2,..,listg}←Item Header Table
for ri in R do
    for aj in A do
        Caj++
for ai in A do
    if Cai<S then
        discard ai;
for ri in R do
    sort Ai in descending order by Caj (aj ∈ Ai) → new ri;
    AddRecordIntoFPTree(new ri);
for listi in List do
    Bai = ExtractConditionalPatternBase(listi);
    Bai = MiningRules(Bai);
    if Bai is single then
        continue;
    else
        goto 21;
```

Frequent pattern tree

This section is devoted to giving a background on the Frequent Pattern Tree algorithm. Preliminary notions related to Frequent Pattern Tree are the following:

- *Attribute*: is an item which could appear in a record. For example, all items in a shopping list can be used as attributes of the shopping record.
- *Record*: is a collection of multiple attributes.
- *Minimum support*: is the minimum number of occurrences that the considered attributes should have.
- *Conditional pattern base*: is the set of all prefix paths ending with the searched element.

Our study used the Frequent Pattern Tree algorithm jar package provided in the open source machine learning software Weka. The process of mining frequent item sets by Frequent Pattern Tree algorithm can be divided into three steps (cf. Algorithm 1):

(1) *Preprocessing of Data set*

(a) Perform the first scan of the data set and count the number of occurrences of each attribute;
(b) Filter the attribute appearing in the data set according to the set minimum support degree and remove the attribute whose number of occurrences is less than the minimum support;
(c) Perform a second scan of the data set and readjust the order of attributes in each record from high to low according to the number of occurrences of attribute.

(2) *Build the Frequent Pattern Tree*

(a) Traverse all records in the data set, starting from the root node, add the corresponding nodes according to the attributes in the records, and add the first occurrence and node entries in the entry header table;
(b) If the node with the same attributes already exists, the count value of the corresponding node is increased. If it exists in the item header table, linked list node is added to the necklace table in the item header table.

(3) *Mining frequent item sets from Frequent Pattern Tree*

(a) Starting from the end of the item header table, for each frequent item (that is, the attribute that meets the minimum support), extract the conditional pattern base from the Frequent Pattern Tree;
(b) Construct the Frequent Pattern Tree with the conditional pattern base of frequent items (the conditional pattern base at this time is equivalent to the record items of the data set);
(c) Repeat steps (a) and (b) until only one path is included.

Methodology

Based on the frequent item set mining algorithm, we propose a botnet detection system comprising the following three parts (cf. Fig. 1).

a) Collection and preprocessing of network data packets;
b) Rules mining using Frequent Pattern Tree algorithm module;
c) Statistical analysis rule set to determine the bot host IP.

Preprocessing of data set

Let us introduce some notions about data processing:

1. *pcap*: is a common file format for storing network packets. Collected packets are stored in a pcap file with a name like xxx.pcap.
2. *arff*: is the specified dataset format that Weka software can read. Below is an example of arff file.

```
@relation weather
@attribute outlook {sunny, overcast, rainy}
@attribute temperature numeric
```

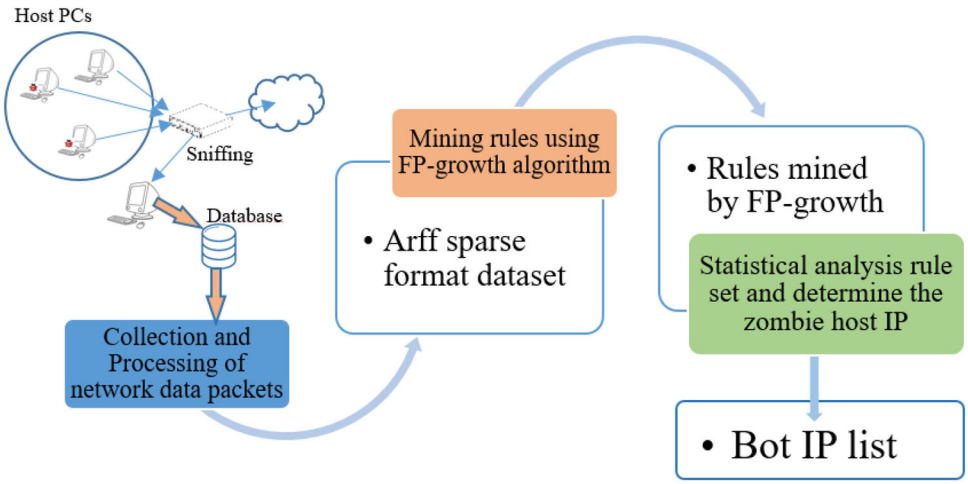

Fig. 1 Scheme of the proposed methodology

```
@attribute humidity numeric
@attribute windy {true, false}
@attribute play {yes, no}
@data
rainy,10,20,true,no
sunny,20,14,false,yes
...
```

The collection of network data packets requires Winpcap library (on Window platform) or Libpcap library (on Linux platform). The Winpcap (Libpcap) library integrates relevant function interfaces for sniffing and collecting network data packets from network adapters (network cards). After sniffing the data packet, the attributes of the data packet are extracted and stored into the specified file according to the arff sparse format specified by Weka. Each network flow corresponds to a record. In this work, the pcap packet files are obtained from the PeerRush [22] dataset (many other pcap datasets are available online): such datasets comprise normal and abnormal network traffic data over a few weeks, where the data have already been preprocessed into a pcap format. It must be remarked that Processing such pcap file into the arff sparse format takes around one day for the PeerRush dataset (which is composed of tens of millions of data). This amount to less than $0.01s$ per data: therefore, when processing data online, this processing time will be so fast that can be neglected.

Mining rules using frequent pattern tree algorithm

The Frequent Pattern Tree algorithm has two important requirements for the data set: the data set must be in a sparse format and the attributes in the data set must be discrete. The processing of the dataset into arff sparse format guarantees that such requirements are satisfied.

Statistical analysis of Rule set

After mining frequent items through the Frequent Pattern Tree algorithm, the data set outputs all the rules that meet the set minimum support. The rule set at this time may contain all kinds of rules that are combined between all characteristic attribute items. In this paper, in view of the need to determine the identity of the host, to ensure that the system can detect the IP of the zombie host that exists in the local area network, so the rule set is first filtered. Because the IP address string format has a special structure, we use regular expression

$$((25[0-5]\|2[0-4]d\|((1d\{2\})\|([1-9]?d))).)$$

$$\{3\}(25[0-5]\|2[0-4]d\|((1d\{2\})\|([1-9]?d)))$$

to filter out all the rule lines that match the IP address string format, and record each independent IP address while counting the independent IP addresses. blackRegular expression is a tool to find strings fitting the pattern set by the user. The pattern under consideration in this work is a pattern to find an IP address. The meaning of the regular expression is the following:

- Standard IP addresses (e.g. 192.162.1.1) must satisfied the rule that each number is between 0 and 255, i.e. $0-255.0-255.0-255.0-255$ (for example, 256.1.1.1 is not an IP address);
- [0−5] indicates one character among 0, 1, 2, 3, 4 (similar for [0-4], [1-9]);
- d represents a one-digit number and dn represents an n-digit number;
- || represent the logical 'or';
- () combines subpatterns into one group;
- ? checks if the char before ? will appear or not: if it appears, it must only appear once (for example, 12?4

Algorithm 2 Statistical analysis of rules

```
Input: Rule set mined by Frequent Pattern Tree algorithm
Output: Bot IP list
Data: arff sparse format dataset D containing records
R←Rule set mined by Frequent Pattern Tree algorithm
Rf←Rule set filtered by IP address regular expression
M=[IP1:C1,IP2:C2,IP3:C3,..]←IP addresses Map with Count
C←The number of related rules to determine whether it is a zombie host IP
BotIP←A list contains bot IP judged by the algorithm
for r in R do
    if r contains IP address then
        Rf.append(r);
        M[r.IP]++;
        if M[r.IP]>C then
            r.IP is a bot IP address;
            BotIP.add(r.IP);
```

includes both 14 and 124, but excludes 1224; $[1-9]?d$ includes both $0-9$ and $10-99$).

Also refer to https://www.cuminas.jp/sdk/regularExpression.html for more details on regular expressions.

When the rule set has been screened and counted, it will enter the judgment. If the rule count value associated with a host's IP address exceeds the threshold C, the host will be judged as a zombie host, issue a warning and record the IP address for the network administrator to conduct subsequent security investigations.

Experiments and evaluation

Dataset description

Our study used the PeerRush Dataset (2018) [22], which is composed of three data sets (see Fig. 2)

1. Normal P2P application data set (with data collected from 11 P2P hosts);
2. Three P2P zombie virus program data sets (with data collected from 13 hosts infected by Storm, 1 host infected by Waledac and 3 hosts infected by Zeus);
3. Non-P2P application data set (with data collected from 42,063 hosts).

The first part of the data packet is generated by a local area network consisting of 11 hosts set up by the PeerRush team. These 11 hosts run 5 popular P2P applications (eMule, μ-Torrent, Frostwire, Vuze, Skype). The application adopts the custom P2P protocol, which ensures the diversification of the P2P application behavior and the underlying mode of the data packet. In order to ensure the human-like characteristics of the data set, the PeerRush team also adopted AutoIt script software to interact with the P2P application to realize the operation of downloading files. The second part of the PeerRush Dataset uses 17 hosts infected by representative P2P botnet virus programs, namely Storm (13 hosts), Waledac (1 host) and Zeus (3 hosts). The third part of the PeerRush Dataset is collected in a separate local area network with 42,063 hosts. In this part of the dataset the use of Snort was combined with existing P2P detection method to filter out suspicious P2P application traffic.

Results and evaluation

The crucial parameter to be selected for the Frequent Pattern Tree algorithm is the minimum support S. A too small minimum support level will mine many worthless rules, which wastes unnecessary calculation time, and might also cause the rule analysis link to become long. On the other hands, if the minimum support level is set too large, it might avoid mining some valuable rules. Our study tested different minimum support in the range [0.001: 0.0005: 0.004] (fractional form): since the total number of examples in the experiment is 33,862,225, this corresponds to [33,862, 50,793, 67,724, 84,655, 101,586, 118,517, 135,448] count types. The fractional form of minimum support means that the number of attribute instances in the total number of instances must be greater than or equal to the minimum support before it is discarded by the algorithm. We use regular expression extraction to experiment with rules related to IP addresses in the mining rule results. The host is identified by IP address, as shown in the results of Table 1.

Since there are 17 bot hosts in total, from the data Table 1, it can be seen that for all values of minimum support, 100% of the zombie hosts (13 hosts infected by Storm, 1 host infected by Waledac and 3 hosts infected by Zeus) are included in the rule results mined by the Frequent Pattern Tree algorithm. It must be also noticed that all the related rule number of each bot IP address are greater than 0, which means that the method blackincludes all the bot hosts; it is the rule number that eventually decides whether it is a bot or not.

However, when the minimum support is low (0.001), 267 rules of normal hosts have been mined (false alarms). More specifically, 14 hosts out of the 11+42,063 hosts are detected as infected: note that there is a rule related to the Normal host with the IP of 139.205.84.245, and there are 19 rules for each of the 14 hosts with 66.154.83. [130–133, 135, 137, 139, 140–142, 145, 147, 150, 152]. blackThe false alarm rate at this time is 46.875% blackwhen the threshold C is set to 10. If C = 10, any IP whose related rules is greater than C, will be considered as a bot IP. As a result, because 19>C the 14 hosts with IP 66.154.83. [130–133, 135, 137, 139, 140–142, 145, 147, 150, 152] are blackconsidered as bots even though they are actually not.

Fig. 2 Bot activities scenario

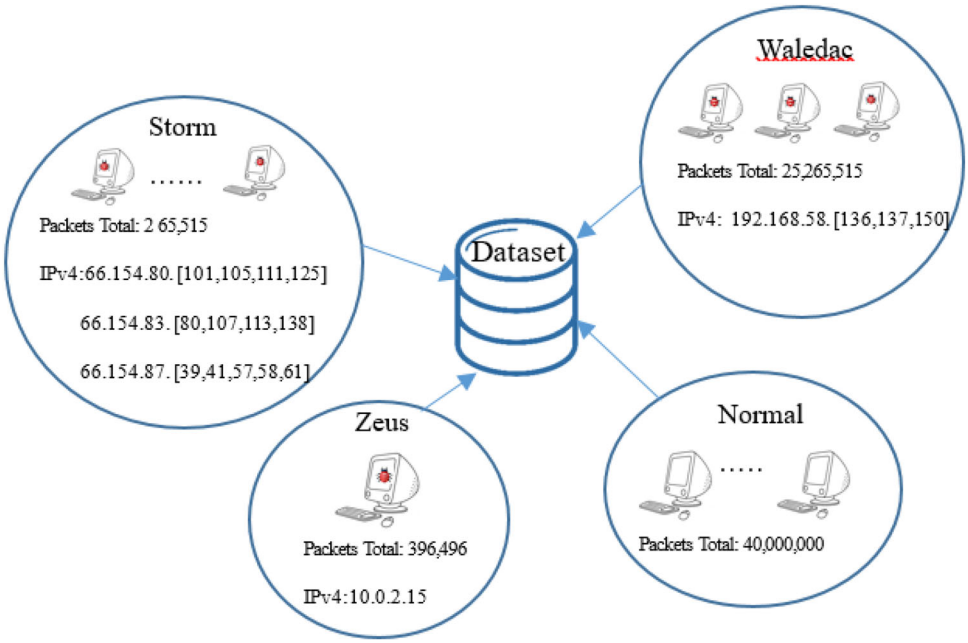

Table 1 Number of detected bot rules and other performance measures

Minimum support	0.001 /33862	0.0015 /50793	0.002 /67724	0.0025 /84655	0.003 /101586	0.0035 /118517	0.004 /135448
Processing time (s)	17	17	16	16	16	16	16
Number of IP Related rules /Total number of rules	606 /868	322 /460	306 /431	273 /394	243 /364	228 /348	227 /347
Activity	Number of detected bot rules						
Storm							
66.154.80.101	20	20	20	15	15	15	15
66.154.80.105	20	20	20	20	15	15	15
66.154.80.111	20	20	20	15	15	15	15
66.154.80.125	20	20	20	15	15	15	15
66.154.83.113	20	20	20	15	15	15	15
66.154.83.107	20	20	20	20	15	15	15
66.154.83.138	32	20	20	15	15	15	15
66.154.83.80	20	20	20	20	15	15	15
66.154.87.39	20	20	20	20	15	15	15
66.154.87.41	20	20	20	20	15	15	15
66.154.87.57	20	20	20	20	15	15	15
66.154.87.58	20	20	15	15	15	15	15
66.154.87.61	20	20	20	15	15	15	15
Zeus							
10.0.2.15	16	11	3	3	3	2	2
Waledac							
192.168.58.136	17	17	16	15	15	10	10
192.168.58.137	17	17	16	15	15	10	10
192.168.58.150	17	17	16	15	15	11	10
Normal							
	267	0	0	0	0	0	0

Table 2 Results with C = 10 and C = 3

Minimum support	0.001 /33,862	0.0015 /50,793	0.002 /67,724	0.0025 /84,655	0.003 /101,586	0.0035 /118,517	0.004 /135,448
C = 10							
Accuracy (%)	99.967	100	99.997	99.997	99.997	99.997	99.997
Precision (%)	54.839	100	100	100	100	100	100
FPR (%)	0.03	0	0	0	0	0	0
FNR (%)	0	0	5.900	5.900	5.900	5.900	5.900
C = 3							
Accuracy (%)	99.967	100	100	100	100	99.997	99.997
Precision (%)	54.839	100	100	100	100	100	100
FPR (%)	0.03	0	0	0	0	0	0
FNR (%)	0	0	0	0	0	5.900	5.900

When blackthe minimum support is greater than 0.001, no false alarms will occur: it is also worth noticing that the number of rules needed to characterize a botnet typically decreases when increasing the minimum support. Except for the Zeus zombie host with an IP of 10.0.2.15, the number of rules is greatly reduced, if the minimum support is greater than blackor equal to 0.0015.

blackTherefore, another parameter that determines the detection effect of the system is the threshold C of the number of independent IP-related rules determined to be a zombie host: when the independent IP-related rule exceeds C, it is determined as a zombie host. The IP is recorded, and then the system issues a warning indicating that a suspicious host is found in the network. We tested C = 10 and C = 3, shown in Table 2. To quantify the performance more accurately, we follow the four commonly used standards for measuring the effectiveness of models in the field of machine learning [19]:

- *Accuracy*: Proportion of correctly judged results (TP + TN) / (TP + FN + FP + TN);
- *Precision*: the proportion of real zombies among all hosts determined as zombies is TP / (TP + FP);
- *False positive rate (FPR)*: the proportion of normal hosts mistakenly judged as zombie hosts FP / (FP + TN);
- *False alarm rate (FNR)*: The ratio of zombies hosts wrongly judged as normal hosts FN / (TP + FN).

The following conclusions can be drawn from the results in Table 2:

1. When the Frequent Pattern Tree mining algorithm support level is set too low, it mines worthless rules containing the IP addresses of normal hosts, resulting in unsatisfactory detection result. Although a zero false alarm rate is guaranteed and no zombie host is spared, many normal hosts are also judged as zombie hosts, resulting in a high false alarm rate (0.4375);
2. The best minimum support is 0.0015, meaning that only attributes with more than 50,000 are considered by the algorithm. This gives the ideal state of 100% accuracy and 100% precision, and there are no missing or false alarms. That is to say, the parameter setting at this time can ensure a greater degree of separation between the botnet host and the normal host, and without losing the rules of the botnet host, as far as possible, limit the occurrence of normal host worthless rules;
3. When the minimum support is further increased, a low false negative situation occurs. After analysis, all this happens because the number of Zeus botnet flow packets in the overall data set is too small compared to other Storm and Waledac, so the strict minimum support makes the Zeus botnet IP related rules reduce below the threshold. Therefore, the choice of minimum support is very important for the detection effect of the detection system designed in this paper.

When C = 3, the proposed method can recognize 100% bot IP without any false alarm if the minimum support is set to 0.0015, 0.002, 0.0025 or 0.003. Our method improves reported results for the same dataset [16]. This is clarified in the comparisons of Table 3 for different values of S. When C = 10, the proposed method has identical results as C = 3 in terms of Accuracy, Precision, FPR and FNR, if the minimum support is set to 0.001, 0.0015, 0.0035, 0.004. The only differences are when the minimum support is set to 0.002, 0.0025, 0.003. In this case C = 10 has same Precision and FPR, but slightly worse accuracy (99.997 % instead of 100%) and higher FNR (i.e. some zombie hosts are wrongly judged as normal hosts). Taking this into account, the most robust minimum support level is 0.0015, because in this case both C = 3 and C = 10 work perfectly.

Table 3 Comparative results

	Malicious	Benign
Decision trees		
Precision (%)	95.3	99.7
Recall (%)	93.4	99.8
FPR (%)	0.2	6.6
Random forest		
Precision (%)	95.3	99.8
Recall (%)	94.9	99.8
FPR (%)	0.2	5.1
Bayesian network		
Precision (%)	91.9	99.5
Recall (%)	88.4	99.7
FPR (%)	0.3	11.6
Proposed ($S = 0.0015$)		
Precision (%)	100	100
Recall (%)	100	100
FPR (%)	0	
Proposed ($S = 0.002$–0.004)		
Precision (%)	100	99.997
Recall (%)	94.117	100
FPR (%)	0	5.882

Conclusion and future work

In this paper, the use of frequent item mining to detect P2P botnets has been investigated. The proposed architecture has shown promising values of accuracy, and the following points can be considered to further improve our investigation. First, the data set used in this article contains only three common P2P botnets, Storm, Zeus, and Waledac. It is of interest to consider more botnet activities possibly including unknown botnets. Second, when selecting attributes, we have considered the characteristics of the source IP address `SrcIP`, source port number `SrcPort`, destination port number `DstPort`, transmission protocol, and the length of the first packet, and it is of interest to consider more attributes (packet number in a flow, bytes in a flow, average packet length in a flow etc.), since a small number of attributes may lose some hidden rules not being mined. Third, we have not considered that the network equipment may change the IP after offline due to the DHCP protocol. In order to address this point, it may be necessary to add a MAC address for dual identity binding in the future.

Compliance with ethical standards

Conflict of interest The authors declare no conflict of interest. No author has a financial or personal relationship with a third party whose interests could be positively or negatively influenced by the article's content. On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Ali I, Ahmed AIA, Almogren A, Raza MA, Shah SA, Khan A, Gani A (2020) Systematic literature review on iot-based botnet attack. IEEE Access 1. https://doi.org/10.1109/ACCESS.2020.3039985
2. Chu Z, Han Y, Zhao K (2019) Botnet vulnerability intelligence clustering classification mining and countermeasure algorithm based on machine learning. IEEE Access 7:182309–182319
3. Dean J, Ghemawat S (2008) Mapreduce: simplified data processing on large clusters. Commun ACM 51(1):107–113
4. Grizzard JB, Sharma V, Nunnery C, Kang BB, Dagon D (2007) Peer-to-peer botnets: Overview and case study. HotBots
5. Hu HX, Wen G, Yu X, Wu ZG, Huang T (2020) Distributed stabilization of heterogeneous mass in uncertain strong-weak competition networks. IEEE Trans Syst Man Cybern Syst 1:1–13. https://doi.org/10.1109/TSMC.2020.3034765
6. Khan WZ, Khan MK, Bin Muhaya FT, Aalsalem MY, Chao H (2015) A comprehensive study of email spam botnet detection. IEEE Commun Surveys Tutor 17(4):2271–2295
7. Kheir N, Han X, Wolley C (2015) Behavioral fine-grained detection and classification of p2p bots. J Comput Virol Hack Tech 11(4):217–233
8. Kumar S, Spafford EH (1994) An application of pattern matching in intrusion detection
9. Li W, Jin J, Lee J (2019) Analysis of botnet domain names for iot cybersecurity. IEEE Access 7:94658–94665
10. Liao WH, Chang CC (2010) : Peer to peer botnet detection using data mining scheme. In: 2010 international conference on internet technology and applications. pp 1–4. IEEE
11. Liu F, Li Z, Nie Q (2009) A new method of p2p traffic identification based on support vector machine at the host level. In: 2009 international conference on information technology and computer science, pp 579–582
12. Masud MM, Gao J, Khan L, Han J, Thuraisingham B (2008) Mining concept-drifting data stream to detect peer to peer botnet traffic. Univ. of Texas at Dallas, Tech. Report# UTDCS-05-08
13. Mathur L, Raheja M, Ahlawat P (2018) Botnet detection via mining of network traffic flow. Proc Comput Sci. 132:1668–1677
14. Meidan Y, Bohadana M, Mathov Y, Mirsky Y, Shabtai A, Breitenbacher D, Elovici Y (2018) N-baiot-network-based detection of iot botnet attacks using deep autoencoders. IEEE Pervasive Comput 17(3):12–22
15. Mythili MS, Shanavas ARM (2013) Performance evaluation of apriori and fp-growth algorithms. Int J Comput Appl

16. Narang P, Hota C, Venkatakrishnan V (2014) Peershark: flow-clustering and conversation-generation for malicious peer-to-peer traffic identification. EURASIP J Inf Secur 2014(1):15
17. Narang P, Ray S, Hota C, Venkatakrishnan V (2014) Peershark: detecting peer-to-peer botnets by tracking conversations. In: 2014 IEEE security and privacy workshops, pp 108–115. IEEE
18. Obeidat AA, Al-Kofahi MM, Bawaneh MJ, Hanandeh ES (2017) A novel botnet detection system for p2p networks. J Comput Sci 13(8):329–336
19. Obeidat AA, Bawaneh MJ (2016) Survey of the p2p botnet detection methods. Int J Emerg Trends Technol Comput Sci (IJETTCS)
20. Pérez MG, Celdrán AH, Ippoliti F, Giardina PG, Bernini G, Alaez RM, Chirivella-Perez E, Clemente FJG, Pérez GM, Kraja E, Carrozzo G, Calero JMA, Wang Q (2017) Dynamic reconfiguration in 5g mobile networks to proactively detect and mitigate botnets. IEEE Internet Comput 21(5):28–36
21. Qiao Y, Yang Yx, He J, Tang C, Zeng Yz (2013) Detecting p2p bots by mining the regional periodicity. J Zhejiang Univ Sci C 14(9):682–700
22. Rahbarinia B, Perdisci R, Lanzi A, Li K (2014) Peerrush: mining for unwanted p2p traffic. J Inf Secur Appl 19(3):194–208
23. Saad S, Traore I, Ghorbani A, Sayed B, Zhao D, Lu W, Felix J, Hakimian P (2011) Detecting p2p botnets through network behavior analysis and machine learning. In: 2011 Ninth annual international conference on privacy, security and trust, pp 174–180. IEEE
24. Torres D (2018) Cyber security and cyber defense for venezuela: an approach from the soft systems methodology. Complex Intell Syst 4:213–226
25. Vormayr G, Zseby T, Fabini J (2017) Botnet communication patterns. IEEE Commun Surveys Tutor 19(4):2768–2796
26. Wang B, Li Z, Tu H, Ma J (2009): Measuring peer-to-peer botnets using control flow stability. In: 2009 International conference on availability, reliability and security, pp 663–669
27. Wang J, Paschalidis IC (2017) Botnet detection based on anomaly and community detection. IEEE Trans Control Netw Syst 4(2):392–404
28. Xia H, Li L, Cheng X, Cheng X, Qiu T (2020) Modeling and analysis botnet propagation in social internet of things. IEEE Internet Things J 7(8):7470–7481
29. Yin L, Luo X, Zhu C, Wang L, Xu Z, Lu H (2020) Connspoiler: Disrupting c c communication of iot-based botnet through fast detection of anomalous domain queries. IEEE Trans Industr Inf 16(2):1373–1384
30. Zhang J, Zhang R, Zhang Y, Yan G (2018) The rise of social botnets: attacks and countermeasures. IEEE Trans Dependable Secure Comput 15(6):1068–1082
31. Zhao D, Traore I, Sayed B, Lu W, Saad S, Ghorbani A, Garant D (2013) Botnet detection based on traffic behavior analysis and flow intervals. Comput Secur 39:2–16
32. Zhuang D, Chang JM (2019) Enhanced peerhunter: detecting peer-to-peer botnets through network-flow level community behavior analysis. IEEE Trans Inf Forensics Secur 14(6):1485–1500