Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Detecting P2P bots by mining the regional periodicity[*]

Yong QIAO[†1], Yue-xiang YANG[1,2], Jie HE[1], Chuan TANG[2], Ying-zhi ZENG[2]

([1]*College of Computer, National University of Defense Technology, Changsha 410073, China*)

([2]*Information Center, National University of Defense Technology, Changsha 410073, China*)

[†]E-mail: qiaoyong10@nudt.edu.cn

Received Feb. 25, 2013; Revision accepted June 18, 2013; Crosschecked Aug. 7, 2013

Abstract: Peer-to-peer (P2P) botnets outperform the traditional Internet relay chat (IRC) botnets in evading detection and they have become a prevailing type of threat to the Internet nowadays. Current methods for detecting P2P botnets, such as similarity analysis of network behavior and machine-learning based classification, cannot handle the challenges brought about by different network scenarios and botnet variants. We noticed that one important but neglected characteristic of P2P bots is that they periodically send requests to update their peer lists or receive commands from botmasters in the command-and-control (C&C) phase. In this paper, we propose a novel detection model named detection by mining regional periodicity (DMRP), including capturing the event time series, mining the hidden periodicity of host behaviors, and evaluating the mined periodic patterns to identify P2P bot traffic. As our detection model is built based on the basic properties of P2P protocols, it is difficult for P2P bots to avoid being detected as long as P2P protocols are employed in their C&C. For hidden periodicity mining, we introduce the so-called regional periodic pattern mining in a time series and present our algorithms to solve the mining problem. The experimental evaluation on public datasets demonstrates that the algorithms are promising for efficient P2P bot detection in the C&C phase.

Key words: P2P botnet detection, Regional periodicity, Apriori, Autocorrelation function, Evaluation function
doi:10.1631/jzus.C1300053 **Document code:** A **CLC number:** TP393.08

1 Introduction

Botnets have become the primary culprits for launching distributed denial-of-service (DDoS) attacks, sending out spam messages, and collecting sensitive information on the Internet. In this paper, we define a botnet as a cluster of compromised hosts, being controlled by unified commands. The owner of the botnet is called the botmaster which sends commands, and the compromised hosts are described as bots for executing the commands unconsciously. Botnets possess remote control functions similar to Trojan horses, and they can automatically propagate like Internet worms. Researchers also find that botnets can be formed by e-mails, social networks (Athanasopoulos *et al.*, 2008), and peer-to-peer (P2P) networks (Starnberger *et al.*, 2008; Nappa *et al.*, 2010).

Botnets are usually classified into three categories by different communication patterns: (1) The Internet relay chat (IRC) botnet. This type of botnet uses IRC channels to issue/receive commands once the bot programs have been planted into compromised hosts. (2) HTTP botnet. In this botnet type, the botmaster utilizes websites to publish commands and the bots will receive commands by HTTP requests. (3) P2P botnet, a novel type of botnet which uses a distributed architecture. The command-and-control (C&C) mechanism within P2P botnets is implemented with P2P protocols, such as Overnet (Grizzard *et al.*, 2007), KAD (Starnberger *et al.*, 2008), and WASTE (Fisher, 2007).

Due to simple structure and easy implementation, IRC botnets have been the most predominant type in the past decade. However, the IRC protocol

[*] Project (Nos. 61170286 and 61202486) supported by the National Natural Science Foundation of China

can be easily recognized by traffic analysis, so hackers utilize HTTP as a botnet C&C mechanism to mix the malicious traffic with regular web browsing. However, both IRC and HTTP botnets suffer from the 'single point of failure', which can be attributed to the centralized control structure. Such a deficiency means that the entire botnet can be destroyed just by targeting the bot controller manipulating the IRC channel, or by shutting down the website where commands are published.

Nevertheless, hackers are also constantly improving their strategies to operate their botnets. In 2007, a P2P-based botnet 'storm' appeared (Holz *et al.*, 2008). It is resilient to defensive countermeasures for traditional botnets as it has no control hub and can survive even if some parts are demolished. Some researchers predicted that P2P-based botnets are the most promising as next-generation botnets (Wang *et al.*, 2010a) and proposed a series of P2P botnet structures, such as the hybrid P2P botnet (Wang *et al.*, 2010b), super P2P botnet (Vogt *et al.*, 2007), and Overbot (Starnberger *et al.*, 2008).

The emergence of P2P botnets has stimulated various detection methods in recent years. Researchers have considered similar methods that work for detecting IRC- or HTTP-based botnets, such as payload inspection, hunting similar traffic between hosts (Gu *et al.*, 2008; Villamarin-Salomon and Brustoloni, 2009; Kang *et al.*, 2011), or machine-learning based methods (Perdisci *et al.*, 2006; Masud *et al.*, 2008; Saad *et al.*, 2011). However, these methods fail in many situations: payload inspection can be evaded by encrypting the traffic; traffic similarity between hosts is not so common in P2P botnets; machine-learning methods cannot easily handle the churn of normal P2P traffic.

In this paper, we propose a novel method for the detection of P2P botnets traffic based on the nature of P2P applications. We investigate some normal P2P applications (such as eMule and BitTorrent) and malicious P2P botnets (such as the Storm botnet and the improved P2P-Zeus botnet). One of our discoveries is that periodicity is an inherent feature through the P2P communication activities, such as periodical search for nodes, periodical finding for buddy-node, and periodical detection for firewalls. Although some non-P2P network applications also exhibit such a feature, like software updating per hour or per day, periodic behavior in P2P is more intensive. Consequently, it is possible to identify P2P botnets traffic by mining such periodic patterns.

However, even though periodicity might be obvious throughout a particular period of time, it is not trivial to identify the activity from the total traffic due to two reasons. First, the time span of the periodic behavior is not fixed. P2P bots can send node requests at various spans, which makes it challenging to locate where the periodic behavior occurs and determine how long the behavior lasts. Second, the period identification can be interfered with by the traffic generated by some other network applications at the same time.

In this study, we build a detection model based on mining the periodic patterns of traffic datasets and propose an evaluation function to distinguish between the non-P2P application traffic, normal P2P application traffic, and malicious P2P botnets traffic. The key contributions from our work can be summarized as follows:

1. We employ regional periodicity as an evaluation criterion to detect P2P botnets traffic and propose a novel detection model named DMRP based on this criterion together with a formal description. To the best of our knowledge, this criterion is novel in P2P botnet detection.

2. We propose three algorithms for mining regional periodic patterns in an event time series.

3. We evaluate the performances of DMRP and three related algorithms using real world network traffic datasets.

2 Related works

2.1 P2P botnet detection techniques

P2P botnets pose severe threats to the security of the Internet. Therefore, this area has received a lot of attention in research communities, and many different concepts and techniques have been proposed for P2P botnet detection. Gu *et al.* (2008) proposed a general botnet detection framework referred to as BotMiner, which is independent of the structure and the protocol of the botnets. BotMiner defines bots as coordinated malware that exhibits similar communication patterns and similar malicious behaviors. Based on this definition, the authors detected P2P bots with similar behaviors and validated the idea using campus network traces. However, it is common that only a small portion of the compromised hosts within a big botnet are being monitored. In such situations, BotMiner does not work well.

Similarly, Noh *et al.* (2009) introduced a P2P botnet detection technique using a multi-phased flow model, in which P2P botnets are identified by observing similar flows occurring between a group of hosts in the network. The two contributions (Gu *et al.*, 2008; Noh *et al.*, 2009) mentioned above and similar techniques adopted in Kang *et al.* (2011) use the similarity property as the key to detecting P2P botnets. These methods are helpful but all require a complete view of the resources that form coordinated hosts, which is not feasible in many real-world scenarios.

Saad *et al.* (2011) studied the abilities of five commonly used machine learning techniques to meet P2P botnet detection requirements. The results showed that it is possible to effectively detect botnets during the botnet C&C phase. But a major limitation for machine learning techniques is that the learning phase needs datasets with a low level of noise, which is hard to satisfy in many cases.

2.2 Behavior detection based on periodicity

Lee *et al.* (2008) proposed a method to search for malicious HTTP botnets by using the degree of periodic repeatability. However, the periodic behavior in this work was identified manually by observation and the authors assumed that the traffic of HTTP requests is pure and not tangled with the traffic of other network applications. Bartlett *et al.* (2011) used a discrete wavelet method to identify low-rate period network traffic and changes in regular communication. This method is helpful in finding the periodic activities with a long-time span, but it cannot locate where the periodic behavior happens or determine the exact span of such behavior. We have studied the periodic patterns in parasite P2P botnets, but the method in our previous work (Qiao *et al.*, 2012) can identify only the existence of periodicity and we applied this method only in parasite P2P botnets. In the current work, we aim to propose novel methods to accurately justify the span, position and degree of periodic behavior in all kinds of P2P botnets.

2.3 Periodicity mining methods

Mining periodic patterns from time series has been studied for a long time. Scientists have proposed many interesting methods to address this problem. For instance, apriori is a famous technique in frequent pattern mining which can significantly reduce the computing cost, stating that (Agrawal and Srikant, 1994): A k-itemset is frequent only if all of its sub-itemsets are frequent. Scientists extended this technique to periodicity mining (Han *et al.*, 1999): each subpattern of a frequent pattern of period p is itself a frequent pattern of period p. This is rational because patterns are stricter than their subpatterns. This principle implies that periodic itemsets can be mined by first scanning the database to find the periodic 1-itemsets, then using the periodic 1-itemsets to generate candidate periodic 2-itemsets, and then checking against the database to obtain the periodic 2-itemsets. This process iterates until no more periodic k-itemsets can be generated for some k. However, this principle is used for mining full periodic patterns firstly, where every point in time contributes (precisely or approximately) to the cyclic behavior of the time series.

Another kind of periodicity looser than full periodicity is partial periodicity (Han *et al.*, 1999; Berberidis *et al.*, 2002), where only some of the points in each periodic segment contribute to the cyclic behavior. Han *et al.* (1999) proposed a very efficient data structure called the max-subpattern hit set and other related algorithms for efficient mining of partial periodic patterns.

Many more methods have been proposed to solve related questions, including mining periodic behavior of object movement (Li *et al.*, 2010), mining partially periodic event patterns with unknown periods (Ma and Hellerstein, 2001), mining asynchronous periodic patterns in time series data (Yang *et al.*, 2003), and mining event periodicity from incomplete observations (Li *et al.*, 2012). These methods can solve problems like the fluctuation of periods, the low efficiency of mining partial periodicity, and the loss of sampling data. However, none of them consider the question of mining the periodic patterns existing in a limited time scope without any prior knowledge. The research in Sheng *et al.* (2006) that proposed to mine dense periodic patterns is closer to our work.

However, the definition of the dense fragment set in Sheng *et al.* (2006) is not suitable for a fast mining algorithm, since its length is not a multiple of the period.

3 P2P botnet detection model: DMRP

3.1 Motivation

P2P bots periodically search nodes to refresh their local peer lists and to receive the latest commands from their botmaster. This is due to the basic demands of the P2P structure and the pre-programmed behavior of bots. Meanwhile, P2P botnets always employ Kademlia-like protocols to construct their own C&C protocols just like popular P2P applications such as eMule (http://www.emule-project.net/) and BitTorrent (http://www.bittorrent.com/). In Kademlia (Maymounkov and Mazieres, 2002), the self-updating mechanism maintains the valid status of applications by periodically sending node search requests to update their node lists in order to obtain closer and more stable neighbors. In P2P botnets, similarly, bots periodically send node search requests to obtain updated commands and neighbors. This phenomenon suggests that if we can capture the specific periodic behaviors in traffic and distinguish them from other activities or other periodic behaviors, we will be able to detect P2P bots.

According to the above analysis, we propose a P2P botnet traffic detection method by mining the regional periodic patterns. 'Regional' indicates that the periodic pattern might exist only within a limited range rather than in the whole time series. A formulated definition of regional periodic pattern will be given Section 4.

Mining potential regional periodic patterns from the time series of traffic features alone is not enough for P2P bot detection. The existence of periodic patterns is not a sufficient condition for the presence of P2P bot traffic, as normal P2P applications and some other network applications may also trigger periodic behavior. Therefore, further steps are needed. In this paper, we introduce an evaluation function to score these mined patterns in order to recognize the P2P bot traffic.

3.2 Detection model

As illustrated in Fig. 1, our P2P bot detection model DMRP (detection by mining regional periodicity) is composed of three major components: time series extraction, regional periodic pattern mining, and evaluation of traffic.

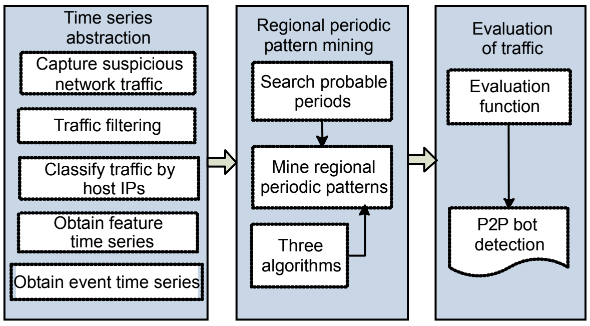

Fig. 1 P2P bot detection model: DMRP (detection by mining regional periodicity)

The workflow of DMRP can be summarized as follows:

1. In the time series extraction phase, to reduce redundant and noisy packets, only suspicious traffic will be retained and then classified by internal hosts' IP addresses. For each IP's traffic, a time series of traffic features will be extracted. At last, based on our later definition of event types, the feature time series will be transformed into an event time series.

2. In the second phase, for an event time series, the circular autocorrelation function is first used to search for candidate periods. Then, as the most important step in our model, we need to recognize regional periodic patterns from current time series based on the candidate periods.

3. In the final phase, the mined regional patterns will be evaluated by an evaluation function. Detection results of P2P bots are determined given the scores of corresponding periodic patterns.

In Section 3.3 we will describe the time series extraction method. The latter two phases about regional periodic pattern mining and the evaluation process will be explained in detail in Sections 4 and 5, respectively.

3.3 Time series extraction

As shown in Fig. 1, the first module of DMRP includes five steps:

1. Capturing suspicious network traffic

Capturing suspicious network traffic needs to be maintained for hours in order to obtain valid data. A long-span large dataset should be split into multiple datasets by day, since the activities of P2P bots like C&C and attacking are barely longer than one day.

2. Filtering

After capturing the suspicious network traffic, redundant packets should be deleted first to reduce noise. Since we focus only on the requests sent by internal hosts, IP packets that are apparently not requests should be discarded first, like incoming packets, domain name system (DNS) packets, and ping packets. Under normal conditions, the payload part in request packets of P2P applications carries only necessary messages, like the specific type of the current packet, the host's node ID, the target's node ID, and some other information. So, it is reasonable to assume that the average packet size of bots' requests falls within a limited small range based on the observation of normal P2P applications. Hence, we set a possible range to filter the packets empirically between 60 and 120 bytes. This step is very important for improving the performance of our detection module as redundant dirty data can largely affect the performance of regional periodic pattern mining.

3. Traffic classification

Our detection module works on internal hosts one by one. A traffic dataset with more than one internal IP address will be split into multiple portions, one for each internal IP.

4. Obtaining time series of traffic features

In this step, we need to choose suitable traffic features to construct a feature time series that can be measured by numerical values and be capable of reflecting the periodic behaviors of P2P botnets during request sending. In addition, the feature candidates should be counted at equal time intervals. We employ the number of packets per second (PPS) to construct the feature time series. Each value at a time point represents the total number of packets during one second. Based on our observation, when P2P applications periodically send requests, there will be a peak of PPS at the first moment. Since P2P applications will continue to send requests to respond to the incoming responses from other nodes, the PPS will be fluctuated for a period of time. There is an obvious fluctuation tendency of PPS at each periodic segment of P2P botnet traffic. That is why we choose PPS as the target feature to represent periodic behaviors.

5. Transformation to event time series

As different periodic segments may not have the same PPS, we cannot directly use the feature time series to run periodicity mining. Here we choose to transfer feature values into event types. Different feature values in a certain range will be regarded as the same event type and the resulting event type can roughly describe the current frequency of sending requests. Specifically, we use single characters, like '*a*', '*b*', '*c*', '*d*', '*e*', to represent different levels of frequencies. Letter '*a*' stands for very few packets per second, while letter '*e*' stands for a very high value of PPS; other letters will represent the intermediate levels of PPSs sequentially. Based on the above transformation rule, a feature time series will be transformed into an event time series, denoted by a string like '*ebcbaabec*…'.

4 Regional periodic pattern mining

4.1 Formulation of regional periodic patterns

In this part, we formally define the concept of regional periodic patterns for event time series. Let S be the set of events (transformed from features) sampled at equal time intervals:

$$S = e_1, e_2, \ldots, e_n,\ \ e_i \in L,\ i \in [1, n], \tag{1}$$

where n is the length of the time series and L is the underlying set of events. Let p be the period of S. Then the number of periods will be $C=\lfloor n/p \rfloor$. So, S can be divided into C segments with equal length and without intersection (the remaining part after C periods in S will be discarded). These segments are called periodic segments and each segment will be

$$E_i = [e_{ip+1}, e_{ip+2}, \ldots, e_{ip+p}],\ \ i \in [0, C). \tag{2}$$

We define a pattern $s=s_1, s_2, \ldots, s_n$ as a non-empty sequence over $(2L-\{\varnothing\})\cup\{*\}$. The wildcard '*' can match any single event from L. Let the L-length of pattern s be the number of s_i which contains letters from L. A pattern with L-length j is also called a j-pattern. For example, a pattern $s=a*c*d$ has three letters which are from L, so we call s a 3-pattern.

Moreover, a subpattern of a pattern $s=s_1, s_2, ..., s_n$ is a pattern $s'=s_1', s_2', ..., s_n'$ such that s and s' are of the same length, and $s_i'=s_i$ or $s_i'=*$.

The frequency function freq(s, S) and confidence function conf(s, S) of a pattern s in a time series S are defined as

$$\text{freq}(s,\ S)=\sum_{i=0}^{C-1}\text{match}(s,\ E_i),\qquad(3)$$

$$\text{conf}(s,S)=\frac{\text{freq}(s,S)}{C},\qquad(4)$$

where $\text{match}(s,E_i)=\begin{cases}1, & \text{if } E_i \text{ matches } s,\\ 0, & \text{else.}\end{cases}$

We say periodic segment E_i matches pattern $s=s_1, \ldots, s_j, \ldots, s_p$ if, for each position j (j=1, 2, …, p), either s_j is * or the letter in s_j equals the jth letter in E_i. Meanwhile, if s' is a subpattern of s and a periodic segment E_i matches s, E_i also matches s'.

Example 1 Set S=*abcdabccabcdabcdabddac* and S has a period p=4. Then S can be divided into five periodic segments E_1=*abcd*, E_2=*abcc*, E_3=*abcd*, E_4=*abcd*, E_5=*abdd*. The last two letters '*ac*' will be dropped. Now if we define string s=*abcd* as a pattern, then freq(s, S)=3, conf(s, S)=3/5=0.6; if we define string s=*ab*d* as a pattern, then freq(s, S)=4, conf(s, S) =4/5=0.8.

Next we will define different periodic patterns based on the concepts described above.

Definition 1 (Full periodic pattern and partial periodic pattern) For a given time series S with length n and an underlying set of values L, we define min_conf as the minimum confidence and min_freq as the minimum frequency count. Assuming p is a period of S:

1. If a pattern $s=s_1, \ldots, s_j, \ldots, s_p$ ($s_i\in L$, $i\in[1, p]$) satisfies

$$\text{conf}(s,S)\geq \text{min\_conf},\qquad(5)$$

$$\text{freq}(s,S)\geq \text{min\_freq},\qquad(6)$$

we say s is a full periodic pattern of S.

2. If a pattern $s=s_1, \ldots, s_j, \ldots, s_p$ ($s_i\in L\cup\{*\}$, $i\in[1, p]$) satisfies

$$\text{conf}(s,S)\geq \text{min\_conf},\qquad(7)$$

$$\text{freq}(s,S)\geq \text{min\_freq},\qquad(8)$$

we say s is a partial periodic pattern of S.

Example 2 Given a time series S=*abcdabccabcd abcdabddac* where S has a period p=4. We define min_conf=0.6 and min_freq=3. Thus, we have $L=\{a, b, c, d\}$. If there is a pattern s=*abcd*, we can see that each letter s_i in s satisfies $s_i\in L$ ($i\in[1,p]$). Based on the analysis in Example 1,

$$\text{conf}(s,S)=0.6\geq \text{min\_conf},\qquad(9)$$

$$\text{freq}(s,S)=3\geq \text{min\_freq},\qquad(10)$$

so pattern s=*abcd* is a full periodic pattern of S. However, if we increase min_conf to 0.8, pattern s will not be a full periodic pattern due to

$$\text{conf}(s,S)=0.6<\text{min\_conf}.\qquad(11)$$

Meanwhile, if there is a pattern s=*ab*d*, then

$$\text{conf}(s,S)=0.8\geq \text{min\_conf},\qquad(12)$$

$$\text{freq}(s,S)=4\geq \text{min\_freq}.\qquad(13)$$

Thus, pattern s=*ab*d* is a partial periodic pattern of S but not a full periodic pattern as s contains '*'.

Definition 2 (Regional periodic pattern) For a given time series S with length n and the underlying set of values L, we define min_conf as the minimum confidence, and min_freq as the minimum frequency count. Here we define a new time series Sub which is a subsequence of S:

$$\text{Sub}=\{E_i,E_{i+1},\ldots,E_j\},\ \ i\geq 0, j\leq p.\qquad(14)$$

We say Sub is a region of S. Assuming p is a period of S and Sub, if a pattern

$$s=s_1,\ldots s_j,\ldots,s_p,\ \ s_i\in L\cup\{*\}, i\in[1,p]\qquad(15)$$

satisfies

$$\text{conf}(s,\text{Sub})\geq \text{min\_conf},\qquad(16)$$

$$\text{freq}(s,\text{Sub})\geq \text{min\_freq},\qquad(17)$$

based on Definition 1, s is a partial periodic pattern of Sub, and here we call s a regional periodic pattern of S.

Example 3 Given a time series S=*abcabcabcabcab dccdadaaaaccbccbccbccac* and period p=3, min_conf =0.6, min_freq=4, if we define a pattern s=*ab**, then $C=\lfloor 37/3\rfloor=12$ and freq(s, S)=5, and thus

$$\text{conf}(s, S)=5/12<\text{min\_conf}, \tag{18}$$

so s is not a partial periodic pattern of S. However, if a subsequence sub_S of S is sub_S=*abcabcabcabcabd*, then we have

$$\text{conf}(s, \text{sub\_}S)=5/5=1>\text{min\_conf}, \tag{19}$$
$$\text{freq}(s, \text{sub\_}S)=5>\text{min\_freq}. \tag{20}$$

Based on Definition 1, pattern s=*ab** is a partial periodic pattern of sub_S. Meanwhile, based on Definition 2, s is a regional periodic pattern of S.

Analysis: The only difference between a full periodic pattern and a partial periodic pattern is that the later can use '*' as part of a pattern. However, both a full and a partial periodic pattern need to calculate the frequency count and confidence for the whole time series. A regional periodic pattern further relaxes the condition of periodic pattern recognition, by requiring only the periodic patterns occur at regional parts of the whole time series. So, a time series may have multiple regional periodic patterns. For instance, in Example 3, '*cc**' is also a regional periodic pattern of S.

4.2 Search for candidate periods

Since we are not clear of the period used in the sending of requests in a given P2P botnet, in order to discover regional periodic patterns in an event time series, we need to determine the period of the time series first. In general, the autocorrelation function (Bracewell and Bracewell, 1986) can be used to find periods in the time series. In our work, we search for possible periods of a time series through the analysis results returned from the autocorrelation function. This method can be explained in the following three steps.

1. Obtaining the binary vector for each event type

Assume a time series

$$S=e_1,e_2,\ldots,e_n,\ \ e_i\in L,\ i\in[1,n], \tag{21}$$

where n is the length of the time series and L is the underlying set of events. We need to create a binary vector of size n for every letter in L. The value of each occurrence of the corresponding letter will be set to 1, and 0 for every other letter. For instance, given a time series S=*abceabccabacabcb*, the binary vector for letter a will be

$$\mathbf{bv}(a,S)=1000100010101000. \tag{22}$$

2. Computing the autocorrelation function

For each binary vector **bv** with a length of N, we calculate the circular autocorrelation function $r(k)$ using the following equation:

$$r(k)=\frac{1}{N}\sum_{x=1}^{N}\mathbf{bv}(x)\cdot\mathbf{bv}(x+k),\ \ k\in[0,N-1]. \tag{23}$$

The average computing complexity for $r(k)$ will be $O(N^2)$, which can be quite expensive, especially when N is a large number. Fortunately, the Wiener-Khinchin theorem (Cohen, 1992) states that "the autocorrelation function of a wide-sense-stationary random process has a spectral decomposition given by the power spectrum of that process". According to this theorem, we can calculate the autocorrelation function by computing the inverse Fourier transform of the power spectrum, which can be directly computed using the Fourier transform of $r(k)$. By means of a fast Fourier transform (FFT) and an inverse fast Fourier transform (IFFT), the computation cost can be reduced to $O(N\log N)$. The processes are shown in the Eq. (24).

Suppose a discrete function $f(x)$=$\mathbf{bv}(x)$, where $F(\omega)$ is the Fourier transform of $f(x)$. Then

$$r(f(x))\xleftarrow{\text{IFFT}}R(F(\omega))\leftarrow F(\omega)\xleftarrow{\text{FFT}}f(x), \tag{24}$$

where $R(F(\omega))=\frac{1}{N}F(\omega)\cdot\overline{F}(\omega)$.

However, in practical applications of the autocorrelation function, the calculation always needs to be adjusted in two ways: first, the vector should deduct the mean value; second, the autocorrelation coefficients need to be normalized in order to easily discover the correlation.

3. Filtering out candidate periods

As shown in Fig. 2, the normalized autocorrelation coefficients of the above binary vector $\mathbf{bv}(a, S)$= 1000100010101000 is given. The first value of the normalized autocorrelation coefficients is the dot product of the binary vector with itself. Since the shifting lag is 0, the value will always be the maximum value of 1. The real peak value is obtained at k=4, which means there is probably a period of 4 and a value of 0.5 at this position, which is an estimation of the degree of correlation. Here we choose 4 as one candidate period. The fact that the following peaks are obtained at positions 8 and 12 (multiples of 4) also verifies the selection of 4 as a possible period.

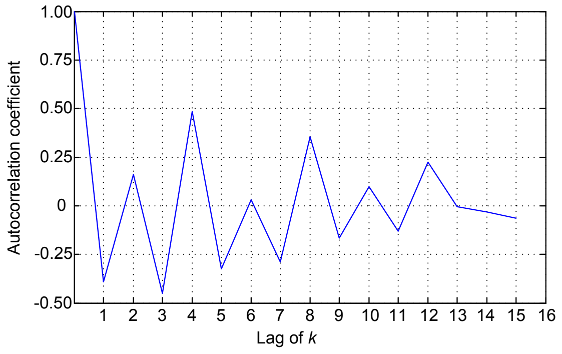

Fig. 2 Normalized autocorrelation function

The above example is employed to explain how the autocorrelation function can be utilized for our detection purpose. In general, we can pick out the candidate periods by two methods as listed below:

1. When searching for periods by analyzing the autocorrelation coefficients artificially, if consecutive local peaks appear at equal intervals, then we can easily identify this interval or the first position of the peak value as a period. As in the previous example, the peaks appear at 4, 8, 12, which is why we choose 4 as the candidate period. In other words, the greatest common divisor (GCD) of the positions of consecutive peaks will be a choice as a candidate period.

2. To identify the period automatically, we first need to find the local peak value with the lowest k value, and then verify whether local peaks appear at positions $2k$ and $3k$ (if they exist). If three consecutive peaks are verified as well, we can directly identify k as a candidate period.

In general, we will obtain multiple periods; some of them might have the same value as the events may be interrelated. Meanwhile, the periodic patterns we want to find here are active only in some regional parts and thus there may be different periods in different regional parts. We will exclude the duplicated candidate periods and then pass them to the next step for regional periodicity mining.

It is a common practice to identify the periods of a time series through a Fourier transform method. Berberidis *et al*. (2002) and Li *et al*. (2010) also used this method to determine periodicity. Our above work further gives an automatic method to search for potential periods.

4.3 Mining algorithms

Traditional methods for periodic pattern mining are not suitable for regional periodic pattern mining, as they are working on the whole time domain. Nevertheless, partial periodic patterns and regional periodic patterns are similar in many aspects as demonstrated above. So, the methods used in partial periodic patterns can be helpful for regional periodic pattern mining with appropriate modifications.

In this section, we propose three different algorithms for regional periodic pattern mining. We introduce a brute-force method based on the classic method for mining partial periodic patterns. Then we introduce two alternative novel methods: apriori-like and RBI (recommendation of regions based on 1-pattern intensities).

Note that in our three algorithms, if a pattern l is a regional periodic pattern in both region A and region B with $A \subset B$, we save only the larger region B.

4.3.1 Brute-force algorithm

Based on the definition of a regional periodic pattern, the regional periodic pattern mining is equivalent to partial periodic pattern mining if we can provide the region of the time series. So, the problem can be transferred to how to provide regions or sub-sequences of the entire time series to be analyzed. A simple way is to use an exhaustive method that enumerates all possible regions in the entire time series, and then use the partial periodicity mining method to search for partial periodic patterns. In this study, we adopt the partial periodicity mining method proposed by Han *et al.* (1999), termed the max-subpattern hit set algorithm (MSHS). Based on Definition 2, mined partial periodic patterns in a certain region of the time series are the regional periodic patterns of the entire time series.

However, this method is computationally intensive. Given a time series with maximum periodic segments of C, and the minimum frequency k=min_freq, the amount of possible subsequences will be

$$\begin{aligned} \text{Num} &= (C-k+1)+\cdots+2+1 \\ &= O((C-k+1)^2). \end{aligned} \tag{25}$$

Compared with C, k is always relatively small. Thus, the total number of possible sub-sequences is approximately $O(C^2)$. Meanwhile, the periodic pattern mining algorithms like MSHS always need complex operations, both in time and space, and the time for running MSHS processes $O(C^2)$ times is not acceptable for practical applications.

4.3.2 Apriori-like algorithm

As described in the related works section, a popular key idea for efficient computation in periodic pattern mining is the apriori property (Han *et al.*, 1999; 2007): each subpattern of a periodic pattern of period p is itself a frequent pattern of period p. This allows us to use periodic patterns of size i as filters for candidate patterns of size i+1. In regional periodic pattern mining, the property supporting the apriori still holds.

Theorem 1 Each subpattern of a regional periodic pattern of period p is itself a regional periodic pattern of period p.

Proof Suppose a pattern s with confidence conf(s, Sub) and frequency freq(s, Sub) (Sub is a region of S) is a regional periodic pattern of time series S with minimum confidence min_conf and minimum frequency min_freq, which means

$$\text{conf}(s,\text{Sub}) \geq \text{min\_conf}, \tag{26}$$

$$\text{freq}(s,\text{Sub}) \geq \text{min\_freq}. \tag{27}$$

For any sub_s which is a subpattern of pattern s, based on the definition of a sub-pattern and considering Eqs. (26) and (27), we can easily infer that

$$\begin{aligned} &\text{conf}(\text{sub\_}s,\text{Sub}) \geq \text{conf}(s,\text{Sub}) \\ &\Rightarrow \text{conf}(\text{sub\_}s,\text{Sub}) \geq \text{min\_conf}, \end{aligned} \tag{28}$$

$$\begin{aligned} &\text{freq}(\text{sub\_}s,\text{Sub}) \geq \text{freq}(s,\text{Sub}) \\ &\Rightarrow \text{freq}(\text{sub\_}s,\text{Sub}) \geq \text{min\_freq}. \end{aligned} \tag{29}$$

According to Definition 2, sub_s is also a regional periodic pattern of S. The proof is completed.

Therefore, based on the improved apriori property in Theorem 1, we propose our apriori-like algorithm for mining regional periodic patterns. The algorithm consists of two steps:

1. Finding the set of regional frequent 1-patterns of period p, F_1

The detailed processes are listed in Algorithm 1.

Algorithm 1 Finding the set of regional frequent 1-patterns of period p

```
Input: $S$, $p$, mc=min_conf, mf=min_freq
Output: $F_1$
1  $E_i=[e_{ip+1},e_{ip+2},\ldots,e_{ip+p}]$, $i\in[0,C)\Leftarrow C=\lfloor |S|/p \rfloor$
2  for $j$←1 to $p$ do
3    $J=[E_{0j},E_{1j},\ldots,E_{(C-1)j}]$ // events in position $j$
4    LJ=distinct($J$)            // distinct letters in $J$
5    for $l$ in LJ do            // $l$ is an event type
6      TP=$J(l)$                 // TP records the time points of $l$ in $J$
7      TS⇐Transfer TP to multi-segments by putting the
           successive time points in one segment
8      for start←1 to LEN=|TS| do
9        for end←LEN to start do     // descending
10         $R$=TS [start to end]      // selective region
11         freq($l$, $R$)⇐the number of events in $R$
12         Span=Endtime(TS)−Begintime(TS)+1
13         conf($l$, $R$)=freq($l$, $R$)/Span
14         if freq($l$, $R$)≥mc and conf($l$, $R$)≥mf do
15           if $R$⊄regions in FJL do
16             FJL+={$R$, conf($l$, $R$)}  // add pairs
17           end               // end line 15
18           go to line 8 for next start
19         end                 // end line 14
20       end                   // end line 9
21     end                     // end line 8
22     FJ+={FJL}               // add FJL to FJ
23   end                       // end line 5
24   $F_1$+={FJ}               // add FJ to $F_1$
25 end                         // end line 2
```

We use a structure F_1 to store the region and confidence for each regional 1-pattern, and then for each position j in the periodic segments and each letter l, we try to check every potential region and justify whether l is a periodic pattern based on conditions like min_conf and min_freq.

In the process of finding F_1, we employ two strategies to reduce the computation cost.

First, we transfer the time points to segments by putting the successive time points in one segment (line 7). This step can efficiently reduce the number of potential regions as the number of segments is much smaller than the number of time points in most cases.

Second, when looping over the regions, we set the end border of one region from the farthest segment to a closer one (line 9). This helps terminate the current loop as soon as we find one region (line 18) to make l a pattern of F_1. This is because if a wider region is verified, there is no need to check the shorter one again.

Based on the above two strategies, we can greatly reduce the computation time spent for looping over the potential regions to find regional 1-patterns. Furthermore, if we have verified a region for current l that satisfies the current conditions, we further check whether the current region is contained in any other region that has already been added to our existing regions (line 15). If yes, we do not submit the current region to assure that the regions returned for current l always have the largest span.

2. Finding all regional periodic i-patterns of period p

Once we have obtained F_1, the set of regional 1-patterns, and the coordinated regions for patterns, we need to find the i-patterns ($i\geq 2$) and terminate immediately when the candidate frequent i-pattern set is empty, based on the idea of apriori.

However, the regional 1-patterns here at each position of periodic segments may cover multiple letters in different regions (Fig. 3a). In the worst case, each letter in L could be a feature in each periodic position. So, if we do not consider the activity ranges for every ($i-1$)-pattern when we create the candidate i-patterns ($i\geq 2$) from ($i-1$)-patterns, we will obtain a gigantic set of candidate i-patterns, which will introduce a huge amount of computational cost.

In our algorithm we consider the coordinated region for every sub-pattern. Two sub-patterns are allowed to compose a candidate pattern with larger L-length only when they have the intersection between their regions. After obtaining the candidate i-patterns, we can use a similar method introduced in Algorithm 1 to check the regional periodicity of each candidate i-pattern. Note that the potential sub-regions are limited to the cross region within the whole time domain (the range between the horizontal lines as shown in Fig. 3a).

4.3.3 RBI algorithm

In this subsection, we propose a new algorithm that quickly recommends regions that may be used to mine for periodic patterns with a large L-length. This algorithm is composed of three steps as listed below:

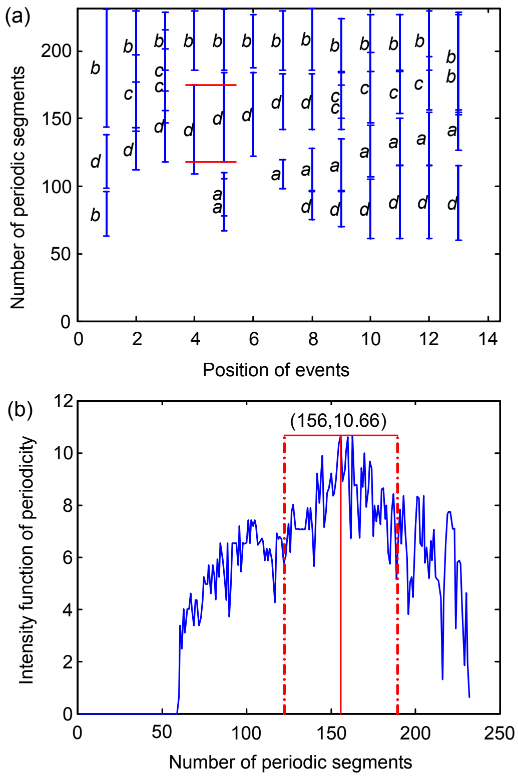

Fig. 3 The regions of 1-patterns and the candidate regions based on intensities of 1-patterns
(a) The 1-pattern regions for event types; (b) The intensities of 1-patterns

1. First, we need to obtain the set of regional periodic 1-patterns F_1 in the time series, similar to the first step in the apriori-like algorithm. Note that the confidence of each 1-pattern is needed and important in the current algorithm.

2. Second, we introduce a function called IFP (intensity function of periodicity), as shown in Eq. (30), which indicates the intensity of periodicity at every time point. The intensity is calculated by summation of the confidence from 1-patterns with coordinated regions that pass the current time point.

$$\mathrm{IFP}(t)=\sum_{s\in F_1}\mathrm{conf}(s,S),\quad t\in \mathrm{Span}(s), \tag{30}$$

where s refers to each pattern from F_1, and Span(s) means the time span of the region of s. Once we obtain the IFP function of every time point t, we need to scan the whole value range and capture the regional peaks. Based on identified local peaks, we can recommend regions that might have a larger probability of containing regional periodic patterns with larger *L*-length.

For our application in detecting P2P botnets, the most important thing is whether the time series has intensive periodicity inside or not. So, this algorithm focuses on mining regions that have a much larger probability of being mined for periodic patterns rather than all potential patterns with their largest regions. Usually, we recommend a region CR:

$$\text{CR} = [\text{Segs(peaks)} - \text{lag},\ \text{Segs(peaks)} + \text{lag}], \quad (31)$$

where $\text{lag} = \dfrac{\text{min\_freq}}{\text{min\_conf}}$, and Segs(peaks) is the ordinal of the segments on which we obtain the local peaks of the IFP function. As seen in Fig. 3b, an example is given to describe the recommended regions.

3. After obtaining the recommended regions, we can use the apriori-like algorithm to mine for the regional periodic patterns within the current candidate region CR.

4.3.4 Algorithm analysis

The brute-force algorithm generates all possible regions first and then uses the MSHS algorithm to obtain the partial periodic patterns, and then all mined partial periodic patterns with corresponding regions constituting the set of regional periodic patterns. The apriori-like algorithm is based on the apriori property and several pruning strategies are introduced to reduce the computational cost. The RBI algorithm further reduces the computational cost by providing local candidate ranges to be analyzed, which is more practical for applications. Brute-force and apriori-like algorithms can give the full sets of regional periodic patterns, while the RBI algorithm focuses on practical applications and provides only parts of regional periodic patterns.

5 Evaluations of traffic

The algorithms proposed in Section 4 can mine for regional periodic patterns in a time series. However, this is not sufficient for detecting the traffic of P2P botnets. Periodic behaviors exist not only in P2P botnets traffic but in many normal P2P applications and some non-P2P network applications. Therefore, we need to distinguish the periodic behaviors of P2P botnets from others.

5.1 Distinguishing periodic behaviors from P2P bots and non-P2P applications

The non-P2P applications show limited periodicity for regular updating or heartbeat detection. However, compared with P2P bots, the degree of periodicity of non-P2P applications is weak under normal conditions. We propose a metric *d*, as seen in Eq. (32), to measure the degree of periodicity of a mined regional periodic pattern *s*:

$$\text{d}(s) = \frac{L\text{\_length}(s)}{p} \cdot \text{max\_conf}(s), \quad (32)$$

where *L*_length(*s*) refers to the *L*_length of pattern *s*, and max_conf(*s*) is the maximum confidence of *s* in any region where *s* meets the conditions of minimum confidence and frequency.

Usually, a periodic pattern represents a type of periodic behavior. Hence, the metric *d* here indicates 'how dense the periodicity such behavior shows is'; e.g., for a full periodic pattern, the *L*_length is equivalent to *p*. For a partial periodic pattern, the *L*_length is always smaller than *p*. So, the full periodic pattern shows denser periodicity if the patterns have the same confidence.

We add a function max_conf(*s*) in Eq. (32) to coordinate with the special situations for regional periodic patterns. It is because, for any regional periodic pattern *s*, there might be multi-regions where *s* is satisfied to be periodic with different confidences. However, here we choose only the region in which *s* shows the maximum confidence.

Since the period *p* of non-P2P applications to repeat periodic behaviors is always large (e.g., Outlook checks new mails every 10–60 min), and the duration for periodic behaviors is not long (e.g., the new mail checking processes last at most 5–20 s), which means *L*_length is small, the regional periodic patterns extracted from traffic of non-P2P applications usually have a much lower *d* than P2P bots. So, *d* can be used as a metric to distinguish the non-P2P applications and P2P bots. This is further discussed in

Section 5.3.

5.2 Distinguishing periodic behaviors from P2P bot traffic and P2P file sharing applications

The period p of P2P file sharing applications (FS-P2P, for short) for sending node-search requests is not as large as that of non-P2P applications; e.g., eMule sends requests to find the owner ID every 60 s by default. However, under normal conditions, this length of period is still much larger than that of the P2P bots we observed (In Storm, the period is 10–15 s). So, we can also use metric d to distinguish the traffic of P2P bots from that of FS-P2P.

However, according to our former research (Qiao *et al.*, 2012) and the analysis of Wang *et al.* (2010a), with the development of P2P botnets, it is possible to create a P2P botnet that entirely uses an existing P2P protocol, like the KAD protocol used in eMule. In this case, the period of sending requests in P2P bots will be exactly the same as the FS-P2P that employs the same protocol, and therefore the metric d will be the same for P2P bots and FS-P2P. However, by combining the messages from hosts and the statistical analysis of periodic activities, it is still possible to identify the malicious usage of public P2P protocols. This is not within the scope of this paper and we focus on P2P botnets with a communication mechanism different from FS-P2P.

5.3 Evaluation function

We propose a metric d to classify P2P bots, FS-P2P, and non-P2P applications. But there are many other network applications that can make unpredictable periodic activities, including on-line games, Ajax web applications, etc. These applications can periodically send requests to the server for updated data in a relatively small interval. Hence, we need more metrics to highlight P2P bots. Based on our observations, under normal conditions only P2P applications will connect to different hosts with varying ports. Other network applications might connect to multiple hosts, but the ports will not change significantly, such as sending HTTP requests to port 80 and sending spam to port 25.

Therefore, we introduce a Boolean function BMP(s) to represent whether the host will connect multiple different ports during the periodic activities, as demonstrated in Eq. (33):

$$\mathrm{BMP}(s)=\begin{cases}1, \text{ if } \sqrt{\dfrac{\mathrm{ports}(s)}{\mathrm{Sum\_pkts}(s)}}\geq 0.4 \mid\mid \mathrm{ports}(s)\geq M,\\ 0, \text{ else},\end{cases} \tag{33}$$

where ports(s) represents the number of distinct ports in the filtered traffic dataset that coordinates within the region of pattern s, and Sum_pkts(s) is the total number of packets in the filtered traffic dataset that coordinates with the region of pattern s.

Meanwhile, our many tests verified that the proportions of the ports(s) and Sum_pkts(s) in P2P bots or FS-P2P applications are always more than 18%. Here we limit the square root of the ratio to be greater than or equal to 0.4. However, the number of available ports for TCP/UDP is limited, the number of packets may increase beyond prediction, and such a situation will make the value ports(s)/Sum_pkts(s) become dramatically reduced. As an extra constraint, we set a threshold M for ports(s) in Eq. (33). When ports(s) is greater than M, then BMP(s)=1; otherwise, BMP(s)=0. The value of M should be set according to the following two principles: (1) The distinct ports that appear during the span of a regional periodic pattern of normal applications cannot reach M or can hardly reach M; (2) The distinct ports that appear during the span of a periodic pattern of P2P traffic can easily reach or go beyond M.

Based on these two principles and empirical practice, we recommend that M be set between 400 and 500. Meanwhile, the specific determination of M is related to the conditions of regional periodicity, such as min_conf(s) and min_freq(s).

Based on the metrics d(s) and BMP(s), we propose our evaluation function Score(S, F, p).

Evaluation function: For a traffic dataset D, S is the extracted time series of behavioral features in D, F is the set of mined regional periodic patterns coordinated, and p is the period. We define the evaluation function Score(S, F, p) as

$$\begin{aligned}&\mathrm{Score}(S,F,p)\\ &=\max_{s\in F}\{d(s)\cdot\mathrm{BMP}(s)\}\\ &=\max_{s\in F}\left\{\frac{L\_\mathrm{length}(s)}{p}\cdot\mathrm{max\_conf}(s)\cdot\mathrm{BMP}(s)\right\}.\end{aligned} \tag{34}$$

As seen in Eq. (34), the value of function Score is within [0, 1]. Therefore, we can use the result of function Score(S, F, p) to judge the classification of different traffic datasets. Here we give a fuzzy criterion in Table 1 for traffic classification and P2P botnet traffic recognition.

Table 1 A criterion for traffic classification based on DMRP

Traffic type	Score range
P2P botnet traffic	(0.3, 1.0]
Fuzzy P2P traffic	(0.2, 0.3]
Normal P2P traffic	(0, 0.2]
Traffic with no periodicity or periodic traffic with BMP=0	0

6 Experiments

We carried out experiments to illustrate the processes of the DMRP model for detecting P2P bots. The performance of the algorithms for regional periodic partial pattern mining was evaluated and scores of the evaluation function for traffic classification were also verified.

6.1 Datasets

The first dataset (Saad *et al.*, 2011) we used here was obtained from the ISOT research lab in the University of Victoria. It is a combination of several publicly available malicious and non-malicious datasets. The malicious datasets contain malicious traffic involving Storm, Waledac, and Zeus botnets, captured by the French chapter of the Honeynet Project.

Among them, Storm botnet is one prevalent P2P botnet, Waledac is an HTTP-based botnet, while Zeus is using HTTP or IRC as its communication mode. The non-malicious datasets are composed of a variety of network activities spanning Web and email to software backup and streaming media. By using the replay and IP mapping technique, the ISOT lab has merged the above two individual trace files into one dataset under one subnet 172.16.0.0/16 (Table 2).

The second dataset we used here was obtained from a campus network which contains live eMule traffic in 1 hour. It has only one Intranet IP address 172.26.75.116 (Table 2).

Table 2 Two datasets used in this study

Source	IP address	Type of traffic
ISOT	172.16.0.11	Waledac & non-malicious
	172.16.0.12	Storm (SMTP spam)
	172.16.2.2	Non-malicious
	172.16.2.3	Non-malicious
	172.16.2.11	Storm & non-malicious
	172.16.2.12	Zeus & non-malicious
	172.16.2.13	Non-malicious
	172.16.2.14	Non-malicious
	172.16.2.111	Non-malicious
	172.16.2.112	Non-malicious
	172.16.2.113	Non-malicious
	172.16.2.114	Non-malicious
Campus	172.26.75.116	eMule

6.2 Dataset pre-processing

1. Filtering

In accordance with the extraction processes in Section 3.3, in the filtering phase of our experiments we set the length range of the interest packet as [60, 120] bytes and split the two datasets into multiple sub-datasets by IP addresses.

Furthermore, we split each IP's dataset whenever the timestamps of two adjacent packets had more than a 1-hour interval. Some sub-datasets were dropped if they were too short for further analysis. In our experiments, we set the minimum length of datasets as 100. The results of sub-datasets are listed in Appendix A.

2. Transforming feature time series to event time series

In accordance with the extraction processes in Section 3.3, here we use 'a', 'b', 'c', 'd', 'e' to represent different levels of frequencies of PPS. We sorted the feature time series in ascending order, and chose the values at 20%, 40%, 60%, 80%, 100% of the series as the upper limits of event types 'a', 'b', 'c', 'd', 'e', respectively. The feature time series can then be converted into event time series.

6.3 Mining for the candidate periods

As shown in Figs. 4 and 5, the autocorrelation analysis results are given for the sub-datasets that can be extracted for candidate periods. Other datasets that cannot provide candidate periods are shown in Appendix B. Table 3 lists the candidate periods extracted.

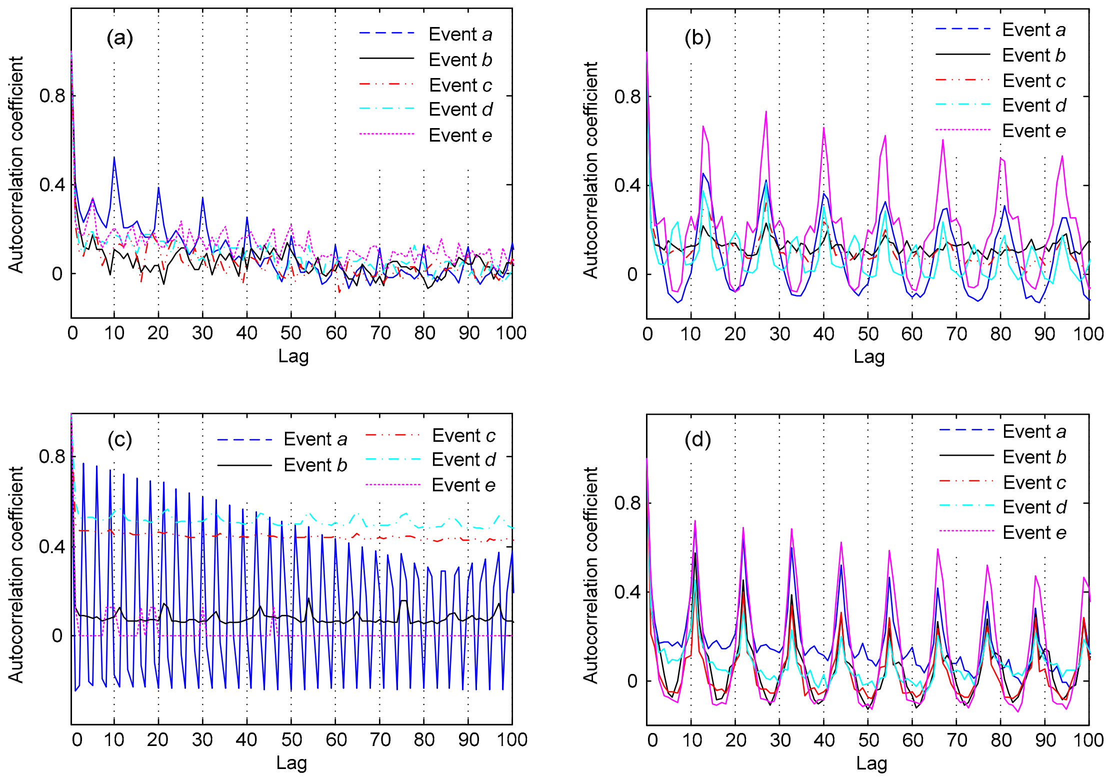

Fig. 4 Autocorrelation analysis of four sub-datasets of the first dataset (only the part of lag≤100 is shown)
(a) 172.16.0.11; (b) 172.16.0.12; (c) 172.16.2.2; (d) 172.16.2.11

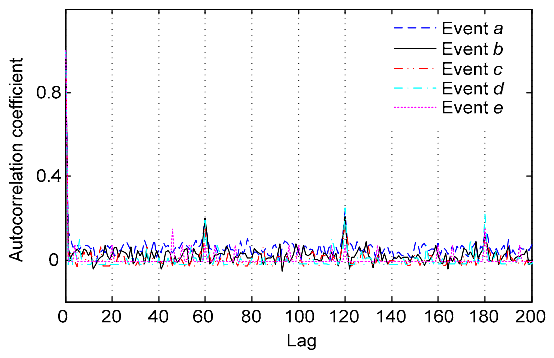

Fig. 5 Autocorrelation analysis of the second dataset for IP 172.26.75.116 (only the part of lag≤200 is shown)

Table 3 The candidate periods

Dataset	IP	Candidate period (s)
1	172.16.0.11	10, 4
	172.16.0.12	13
	172.16.2.2	3
	172.16.2.11	11
2	172.26.75.116	60

According to our basic hypothesis, traffic (as seen in Appendix B) that does not show periodicity at all in autocorrelation analysis is almost impossible for P2P applications, so we dropped the analysis of such datasets of traffic in the following steps.

6.4 Performance of algorithms

1. Feasibility

In our experiments, we set the minimum confidence of regional periodic patterns as min_conf=0.65, and the minimum frequency of patterns as min_freq=20. As seen in Table 4, the brute-force algorithm and apriori-like algorithm can successfully dig out all regional periodic patterns that satisfy the conditions in the time series. Since the brute-force algorithm uses loops to check all possible regions and mine for the needed patterns, its result is considered as comprehensive. The apriori-like algorithm is able to lead to the same result as the brute-force algorithm, which indicates that the apriori property used in our apriori-like algorithm is rational. The RBI algorithm does not obtain the complete set of patterns. However, as seen in Table 4, at any level of *L*-length, the results of RBI are not absent, which shows that RBI works in most cases in which long periodic patterns are needed. In other words, RBI is able to find the regions in the traffic with the densest periodicity.

Table 4 Results of regional periodic pattern mining

IP	*L*_length	Number of patterns	
		Brute-force (apriori-like)	RBI
172.16.0.11 (*p*=10)	1	2	2
172.16.0.11 (*p*=4)	1	0	0
172.16.0.12	1	46	46
	2	84	60
	3	84	69
	4	43	37
	5	8	7
172.16.2.2	1	83	83
	2	48	3
	3	6	1
172.16.2.11	1	29	29
	2	48	45
	3	47	46
	4	30	30
	5	12	12
	6	2	2
172.26.75.116	1	31	31
	2	46	45
	3	30	30
	4	8	8
	5	1	1

2. Efficiency

As shown in Table 5, the computation time for the brute-force algorithm is always long, which indicates that the brute-force algorithm can be used only in theoretical analysis but not in practical applications. The apriori-like algorithm and RBI algorithm are promising as both of them can complete the mining processes in a short time.

Table 5 Time cost of the three algorithms

IP	Time (s)		
	Brute-force	Apriori-like	RBI
172.16.0.11 (*p*=10)	7.52	0.21	0.19
172.16.0.11 (*p*=4)	56.32	0.43	0.22
172.16.0.12	466.05	5.19	4.11
172.16.2.2	>1800	452.76	16.17
172.16.2.11	136.26	2.82	1.63
172.16.75.116	2.64	2.71	1.38

However, there are two irregular situations in Table 5. First, for mining patterns in the time series of IP 172.26.75.116, the brute-force algorithm is able to quickly complete the process. This is because the period of the current time series is large (60 s), and thus the periodic segments here are very small, which means the loop time of the brute-force algorithm is small, and eventually this mining process can be done quickly.

Second, when processing the IP 172.16.2.2, the apriori-like algorithm costs 452.76 s to complete the mining process, which is because the period of the time series of the current IP is too small (*p*=3), with a relatively small frequency of pattern (min_freq=20), the set of 1-patterns is very large, and the combination for high *L*-length patterns is computationally intensive. In such cases, using the RBI algorithm can significantly reduce the computational cost (Table 5). Based on the above analysis, we learned that the brute-force algorithm is not feasible in practice, while the apriori-like algorithm and RBI algorithm are efficient in mining for regional periodic patterns in most cases.

6.5 P2P bot detection

The evaluation scores are listed in Table 6. Even though the set of regional patterns of the RBI algorithm is not complete, the scores from different algorithms do not exhibit obvious differences for identification of the traffic type. As seen in Table 6, if we do not consider the effect of the BMP function, the patterns with a maximum of *d*(*s*) from different algorithms are nearly the same for every IP address. The reason is that for every high level of *L*-length from the brute-force or apriori-like algorithm, RBI obtains at least one pattern with this level of *L*-length. This means even RBI cannot obtain the exact pattern with a maximum of *d*(*s*). Different patterns with the same *L*-length will not greatly affect the final evaluation scores.

For example, for the traffic of 172.16.0.12, if the BMP is not false here, the score from the RBI algorithm will be $5/13\times0.8\approx0.308$ and from the brute-force or apriori-like algorithm will be $5/13\times0.916\approx0.352$. These two scores should be classified into the same class as shown in Table 1. The final scores are both 0, however, since the BMP for 172.16.0.12 is false.

Table 6 Evaluation scores for traffic datasets[*]

Algorithm	IP	*p*	Region[**]	*L*_length	Max_conf	Port	Sum_pkt	BMP	Score
Brute-force, Apriori-like	172.16.0.11-1	10	34–61	1	0.714	190	21 831	0	0
	172.16.0.11-2	4	None						0
	172.16.0.12	13	**132–155**	**5**	**0.916**	**3**	**12 934**	**0**	**0**
	172.16.2.2	3	34–63	3	1.000	2	1693	0	0
	172.16.2.11	11	58–79	6	0.909	1184	22 475	1	0.496
	172.26.75.116	60	6–29	5	0.833	471	868	1	0.069
RBI	172.16.0.11-1	10	34–61	1	0.714	190	21 831	0	0
	172.16.0.11-2	4	None						0
	172.16.0.12	13	**130–154**	**5**	**0.800**	**3**	**13 117**	**0**	**0**
	172.16.2.2	3	34–63	3	1.000	2	1693	0	0
	172.16.2.11	11	58–79	6	0.909	1184	22 475	1	0.496
	172.26.75.116	60	6–29	5	0.833	471	868	1	0.069

[*] min_conf=0.65, min_freq=20, *M*=500. [**] Listed by the ordinal of the periodic segment

The scores for IPs 172.16.0.11 and 172.16.2.2 are also calculated as 0 since BMP is false. Now, only two IPs 172.16.2.11 and 172.26.75.116 do not obtain their scores. Based on the scores and the classification criterion in Table 1, we identify that the host of IP 172.16.2.11 is most likely to contain P2P bot traffic with the score 0.496>0.3, and that IP 172.26.75.116 probably contains only normal P2P traffic for the score 0.069<0.2.

Compared with the actual classifications as seen in Table 2, our judgment about IP 172.16.2.11 containing P2P botnet traffic is correct, and the judgement about IP 172.26.75.116 is also correct. The dropped datasets in our analysis are also verified to assure that they do not have P2P botnets traffic. Meanwhile, our method can precisely tell the position and duration of the malicious periodic behaviors. For example, for IP 172.16.2.11, one of the mined regional periodic patterns with *L*_length=6 is s= $c{*}e{*}aab{*}{*}{*}c$, and the region of s is $[E_{58}, E_{59}, ..., E_{89}]$. As the period here is 11, we can infer that the time span is from (58−1)×11=627 s to 89×11=979 s. As seen in Fig. 6, the periodic behaviors in such a region are obvious, and the trend of PPSs changing from high to low and to high again matches the pattern $s=c{*}e{*}aab{*}{*}{*}c$ well.

However, there is some confusion about the evaluation of the host of IP 172.16.0.12. Based on the introduction from Table 2, IP 172.16.0.12 corresponds to Storm botnet, which is a P2P bot, but in the above experiments we do not recognize this IP as a malicious one. This is because our proposed detection

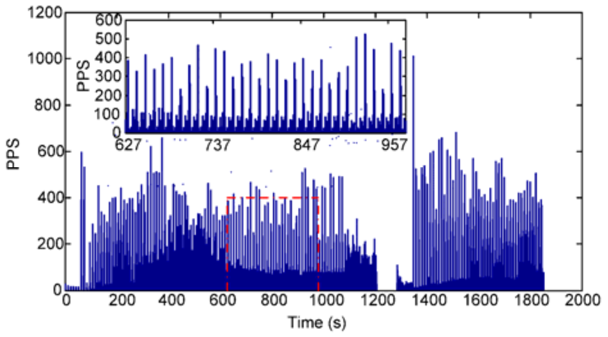

Fig. 6 The time series of the number of packets per second (PPS) for IP 172.16.2.11

model DMRP is based on the properties that exist only in the C&C phase of P2P botnets. Though traffic of IP 172.16.0.12 shows strong periodicity in spamming processes, such periodicity is not essential and can be changed easily by attackers. Nevertheless, in practical detection processes, network traffic with strong periodicity is always suspicious and should be picked out for detailed artificial analysis.

6.6 Comparison analysis

Saad *et al.* (2011) proposed a machine learning method to classify the traffic in the ISOT botnet dataset, which is the same as our dataset 1. The machine learning method can classify the traffic into different classes. Unlike our method, however, it cannot recognize some specific classes. As for the detection of the host with P2P bots, no training is needed using our methods. The time consumption in machine learning is huge: using the support vector machine (SVM) in Saad *et al*. (2011), the classifier training time is 1400 s, and the classification time is 18 s; in contrast, only 16 s is needed using the RBI algorithm in our work. As the P2P bots in our datasets have no similarity between different hosts, similarity-based methods like BotMiner (Gu *et al*., 2008) cannot detect Storm bots or other bots like Waledac and Zeus. Obviously, detection based on the inherent features of P2P applications in our work has the advantage of directly detecting the P2P bots in the C&C phase.

7 Conclusions

In this paper, we have proposed a novel detection model called DMRP to detect P2P botnets. Unlike the existing methods using similarity of network behaviors, or classifying techniques based on machine learning methods, our detection model is based on the nature of P2P botnets, sending requests periodically for updating their peer lists or receiving the commands from botmasters in the C&C phase. Therefore, we identify P2P botnet traffic by analyzing the specific periodicity within traffic datasets.

During the process to mine for hidden periodicity, we introduce a novel concept of regional periodic pattern mining in time series, which is novel to P2P botnet detection, and present three algorithms to solve this problem. The brute-force algorithm is proposed for only theoretical analysis; the apriori-like algorithm and RBI algorithm are very efficient and effective in practical applications.

Experimental evaluation based on public datasets shows that our proposed algorithms are suitable for mining the hidden regional periodicity in time series and that the DMRP model can detect P2P botnet traffic easily and correctly.

However, as our P2P botnet detection model is built on the basic nature of P2P protocols in the C&C phase, it is not able to detect P2P botnets in the attack phase or other types of botnets with different communication protocols, such as IRC and HTTP.

In the future, we aim to improve our detection model in two aspects. The first is to provide better noise-filtering techniques for obtaining a better time series of traffic features, and the second is to build a standard evaluation criterion for traffic classification based on more analysis of different traffic datasets.

Acknowledgements

We thank Dr. Sherif SAAD from the University of Victoria and Dr. Wei LU from Keene State College for providing the datasets and thank Dr. Chengku WU and Dr. Geraint DUCK from Manchester University for polishing the paper.

References

Agrawal, R., Srikant, R., 1994. Fast Algorithms for Mining Association Rules. Proc. 20th Int. Conf. on Very Large Data Bases, p.487-499.

Athanasopoulos, E., Makridakis, A., Antonatos, S., Antoniades, D., Ioannidis, S., Anagnostakis, K.G., Markatos, E.P., 2008. Antisocial networks: turning a social network into a botnet. *LNCS*, **5222**:146-160. [doi:10.1007/978-3-540-85886-7_10]

Bartlett, G., Heidemann, J., Papadopoulos, C., 2011. Low-Rate, Flow-Level Periodicity Detection. IEEE Conf. on Computer Communications Workshops, p.804-809.

Berberidis, C., Aref, W.G., Atallah, M., Vlahavas, I., Elmagarmid, A.K., 2002. Multiple and Partial Periodicity Mining in Time Series Databases. European Conf. on Artificial Intelligence, p.370-374.

Bracewell, R.N., Bracewell, R., 1986. The Fourier Transform and Its Applications. McGraw-Hill, New York.

Cohen, L., 1992. Convolution, filtering, linear systems, the Wiener-Khinchin theorem: generalizations. *SPIE*, **1770**: 378-393. [doi:10.1117/12.130944]

Fisher, D., 2007. Storm, Nugache Lead Dangerous New Botnet Barrage. Available from SearchSecurity.com.

Grizzard, J.B., Sharma, V., Nunnery, C., Kang, B.B.H., Dagon, D., 2007. Peer-to-Peer Botnets: Overview and Case Study. Proc. 1st Conf. on 1st Workshop on Hot Topics in Understanding Botnets, p.1.

Gu, G., Perdisci, R., Zhang, J., Lee, W., 2008. BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection. USENIX Security Symp., p.139-154.

Han, J., Dong, G., Yin, Y., 1999. Efficient Mining of Partial Periodic Patterns in Time Series Database. Proc. 15th IEEE Int. Conf. on Data Engineering, p.106-115.

Han, J., Cheng, H., Xin, D., Yan, X., 2007. Frequent pattern mining: current status and future directions. *Data Min. Knowl. Discov.*, **15**(1):55-86. [doi:10.1007/s10618-006-0059-1]

Holz, T., Steiner, M., Dahl, F., Biersack, E., Freiling, F., 2008. Measurements and Mitigation of Peer-to-Peer-Based Botnets: a Case Study on Storm Worm. USENIX Workshop on Large-Scale Exploits and Emergent Threats, p.1-9.

Kang, J., Song, Y.Z., Zhang, J.Y., 2011. Accurate detection of peer-to-peer botnet using multi-stream fused scheme. *J. Networks*, **6**(5):807-814. [doi:10.4304/jnw.6.5.807-814]

Lee, J.S., Jeong, H.C., Park, J.H., Kim, M., Noh, B.N., 2008. The Activity Analysis of Malicious HTTP-Based Botnets Using Degree of Periodic Repeatability. Int. Conf. on Security Technology, p.83-86. [doi:10.1109/SecTech.2008.52]

Li, Z., Ding, B., Han, J., Kays, R., Nye, P., 2010. Mining Periodic Behaviors for Moving Objects. Proc. 16th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.1099-1108. [doi:10.1145/1835804.1835942]

Li, Z., Wang, J., Han, J., 2012. Mining Event Periodicity from Incomplete Observations. Proc. 18th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, p.444-452. [doi:10.1145/2339530.2339604]

Ma, S., Hellerstein, J.L., 2001. Mining Partially Periodic Event Patterns with Unknown Periods. Proc. 17th IEEE Int. Conf. on Data Engineering, p.205-214. [doi:10.1109/ICDE.2001.914829]

Masud, M.M., Gao, J., Khan, L., Han, J., Thuraisingham, B., 2008. Mining Concept-Drifting Data Stream to Detect Peer to Peer Botnet Traffic. Available from http://www.utdallas.edu/mmm058000/reports/UTDCS-05-08.pdf.

Maymounkov, P., Mazieres, D., 2002. Kademlia: a Peer-to-Peer Information System Based on the XOR Metric. Peer-to-Peer Systems, p.53-65. [doi:10.1007/3-540-45748-8_5]

Nappa, A., Fattori, A., Balduzzi, M., Dell'amico, M., Cavallaro, L., 2010. Take a Deep Breath: a Stealthy, Resilient and Cost-Effective Botnet Using Skype. *In*: Kreibich, C.J.M. (Ed.), Detection of Intrusions and Malware, and Vulnerability Assessment. Springer Berlin Heidelberg, p.81-100. [doi:10.1007/978-3-642-14215-4_5]

Noh, S.K., Oh, J.H., Lee, J.S., Noh, B.N., Jeong, H.C., 2009. Detecting P2P Botnets Using a Multi-phased Flow Model. IEEE 3rd Int. Conf. on Digital Society, p.247-253. [doi:10.1109/ICDS.2009.37]

Perdisci, R., Gu, G., Lee, W., 2006. Using an Ensemble of One-Class SVM Classifiers to Harden Payload-Based Anomaly Detection Systems. IEEE 6th Int. Conf. on Data Mining, p.488-498. [doi:10.1109/ICDM.2006.165]

Qiao, Y., Yang, Y., He, J., Liu, B., Zeng, Y., 2012. Detecting parasite P2P botnet in eMule-like networks through quasi-periodicity recognition. *LNCS*, **7259**:127-139. [doi:10.1007/978-3-642-31912-9_9]

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. IEEE 9th Annual Int. Conf. on Privacy, Security and Trust, p.174-180. [doi:10.1109/PST.2011.5971980]

Sheng, C., Hsu, W., Lee, M.L., 2006. Mining Dense Periodic Patterns in Time Series Data. Proc. 22nd Int. Conf. on Data Engineering, p.115. [doi:10.1109/ICDE.2006.97]

Starnberger, G., Kruegel, C., Kirda, E., 2008. Overbot: a Botnet Protocol Based on Kademlia. Proc. 4th Int. Conf. on Security and Privacy in Communication Networks, Article 13. [doi:10.1145/1460877.1460894]

Villamarin-Salomon, R., Brustoloni, J.C., 2009. Bayesian Bot Detection Based on DNS Traffic Similarity. Proc. ACM Symp. on Applied Computing, p.2035-2041. [doi:10.1145/1529282.1529734]

Vogt, R., Aycock, J., Jacobson, M.J.Jr, 2007. Army of Botnets. Network and Distributed System Security Symp., p.111-123.

Wang, P., Aslam, B., Zou, C.C., 2010a. Peer-to-Peer Botnets. *In*: Handbook of Information and Communication Security. Springer, p.335-350.

Wang, P., Sparks, S., Zou, C.C., 2010b. An advanced hybrid peer-to-peer botnet. *IEEE Trans. Depend. Secur. Comput.*, **7**(2):113-127. [doi:10.1109/TDSC.2008.35]

Yang, J., Wang, W., Yu, P.S., 2003. Mining asynchronous periodic patterns in time series data. *IEEE Trans. Knowl. Data Eng.*, **15**(3):613-628. [doi:10.1109/TKDE.2003.1198394]

Appendix A: Sub-datasets of datasets 1 and 2

Table A1 Sub-datasets of datasets 1 and 2

IP address	Sub-dataset (length)	Valid number*
172.16.0.11	1 (761)	1 (No. 1)
172.16.0.12	1 (3117)	1 (No. 1)
172.16.2.2	1 (64 985)	1 (No. 1)
172.16.2.3	1 (60)	0
172.16.2.11	1 (42), 2 (1852)	1 (No. 2)
172.16.2.12	1 (58), 2 (44), 3 (38 805)	1 (No. 3)
172.16.2.13	1 (5397)	1 (No. 1)
172.16.2.14	1 (3202)	1 (No. 1)
172.16.2.111	1 (41 201)	1 (No. 1)
172.16.2.112	1 (86 360)	1 (No. 1)
172.16.2.113	1 (86 369), 2 (30 588)	2 (No. 1, No. 2)
172.16.2.114	1 (5537)	1 (No. 1)
172.26.75.116	1 (3605)	1 (No. 1)

* The No. 1/2/3 in the brackets refer to the ordinal numbers of the datasets (in the second column) that satisfy the requirement of length

Appendix B: Autocorrelation coefficients for datasets in which periods cannot be extracted

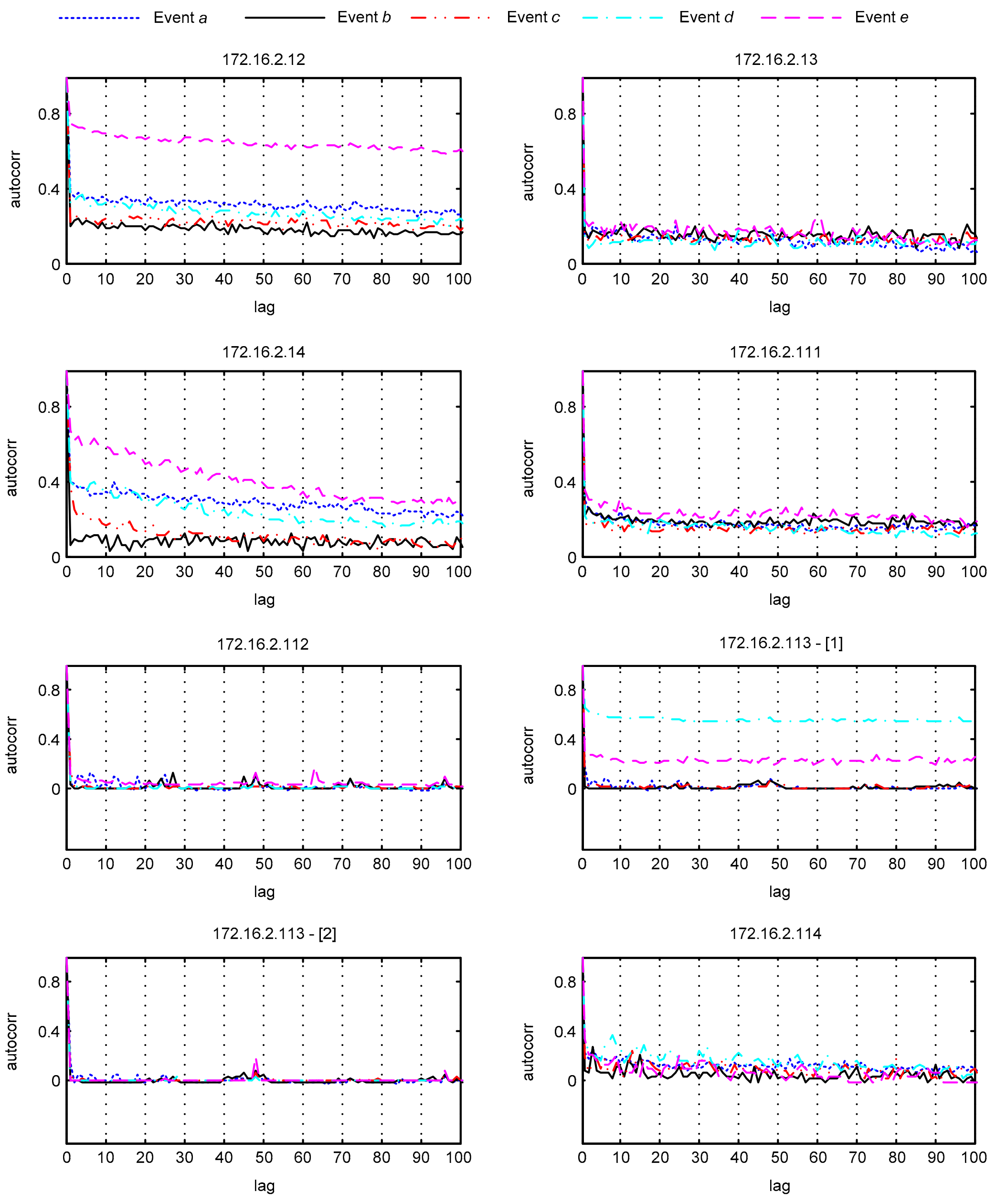

Fig. B1 Autocorrelation results of the remaining eight sub-datasets that show no obvious periodicity (only the part of lag≤100 is shown)