Monsieur Poirot: Detecting Botnets Using Re-Identification Algorithm and Nontrivial Feature Selection Technique

Wei-Min Lee, Amir Rezapour, and Wen-Guey Tzeng
Department of Computer Science, National Chiao Tung University, Taiwan, 30010
Email: wg4i9n.cv98@nctu.edu.tw, rezapour@cs.nctu.edu.tw, wgtzeng@cs.nctu.edu.tw

***Abstract*—Modern botnets are progressively migrating to P2P network to resist against take-down attempts. In addition, new botnets use randomization in their behavior to evade detection. In this paper, we propose a new method for detecting stealthy P2P bots. We formulate the problem as a re-identification problem. This opens the possibility of powerful instantiations of detection algorithms to address the botnet detection problem. We also use a nontrivial feature selection technique to discover the best feature pairs for conducting comparison between two flows. We use real-world botnet data to evaluate the performance of Monsieur Poirot and compare it with existing flow-based algorithms. Monsieur Poirot is robust towards injection of noise in the communication patterns. The experimental results show that Monsieur Poirot is able to identify P2P bots with an average TPR of 98.65% and an average FPR of 0.21%.**

I. Introduction

A botnet is a group of compromised hosts (bots) that are remotely commanded by an attacker (the botmaster) through a command and control (C&C) channel. Botnets are one of the most vital threats to the Internet. Bots serve as an army for a variety of cyber crimes, such as spamming, DDoS attacks, password stealing, click fraud, etc.

Recently, their variants have been increasing towards emergence of defense techniques, including antivirus software, IDS, etc. However, recent studies have shown that with the complex behavior of new botnets, detecting them has become more difficult [1]. For instance, Storm botnet and its updated version Waledac botnet have migrated to the P2P architecture. In a P2P botnet, bots are connected to each other and act as both of C&C servers and clients. Although setting up P2P communication is more costly and may suffer from higher latency, it is not subject to a single point of failure.

为什么p2p结构会使得检测更加难？

Most of previous research for detecting botnets relies on flow-based approaches and machine learning techniques [2], [3], [4], [5]. The detection models are created using botnet datasets which were captured from real interaction among bots. Since bots can evade detection by randomizing their behaviors, the detection models need to be robust against bot randomization. We aim to deal with this issue in our work and build a detection system Monsieur Poirot.

The contributions of our approach are three-fold. First, we formulate the problem of botnet detection as a re-identification problem. We re-identify bots by obtaining information from their network flows. What makes re-identification possible is that feature values are not given randomly, but according to complex behaviors of bots that are not known in advance.

Second, we use real-world datasets of malicious and benign traffic to compare the performance of Monsieur Poirot with flow-based algorithms. It shows that our re-identification technique is more robust in existence of noises. Monsieur Poirot can effectively detect malicious traffic with highest true positive rate (TPR), accuracy (ACC) and precision, while attaining lowest false positive ratio (FPR). Its TPR is 98.65% and FPR is 0.21%, which is twice less than that of the algorithm [5].

Third, our similarity function utilizes a novel *nontrivial feature selection* technique to discover the best feature pairs for measuring similarity between two flows. In a nutshell, it randomly picks k feature pairs which have closest distance in both flows. The experimental results verify the success probability of our nontrivial feature selection technique.

This research is partially supported by MOST project 104-2221-E-009-112-MY3, Taiwan.

II. Related Work

Recent research on P2P botnets has turned to analyze behaviors of network traffic [3], [4], [5], [2]. Some research detected P2P botnets in existence of benign traffic [7], [2].

PeerRush [2] utilized supervising and unsupervised techniques to detect malicious and benign P2P applications. However, its detection accuracy is moderate. Beigi et al. [3] evaluated different sets of features using their feature selection algorithm. Saad et al. [4] conducted a study on P2P botnet detection using five machine learning techniques: SVM, ANN, Nearest Neighbors, Gaussian, and Naive Bayes classifiers. Zhao et al. [5] proposed a detection algorithm using the decision tree with Reduced Error Pruning algorithm (REP-Tree). Our implementation verifies high accuracy in detecting P2P botnets. However, their method had non-negligible false positive ratios.

non-negible不可以忽视的，false positive ratios？（虚报率，误报率）？？、什么证据？

III. System overview

We chop the stream of live network traffic for each host H_l into a *time interval window* W, e.g., 15 minutes. Then flows are extracted. At the end of each time interval, we extract statistical features from flows of node H_l. A database is a two-dimensional matrix, where each row is a network flow from node H_l and each column is a statistical feature extracted from the observed network flow.

We observe the following behaviors of P2P botnets.

- Bots are programmed to periodically engage in P2P communication with servers or other bots to maintain availability and capability of the botnet. Such a periodic communication is necessary for a botmaster to keep pace with available bots and conduct an attack. Due to this behavior, a set of carefully selected features can identify the infected hosts.
- Bots are equipped with an evading method to elude detection. For example, a comprehensive study of Nugache botnet [8] confirms the use of frequent switch between TCP & UDP protocols, port randomization, and arbitrary delay throughout the communications using Sleep() calls [1]. Therefore, most of network flows of the same bot are not similar. We need dynamic selection of features to deal with this problem.

A. Notation

An uppercase boldface letter represents flow $\mathbf{F} = \{f_1, f_2, \ldots, f_n\}$ which consists of n features. $\mathcal{C}_{\mathbf{FP}}$ represents a cluster of flows with center $\mathbf{FP}$[2]. Let $\Delta = \{\mathbf{FP_1}, \mathbf{FP_2}, \ldots, \mathbf{FP_N}\}$ be a set of cluster centroids. $\Delta^j \subset \Delta$ denotes a subset of type j P2P application. We treat network traffic files, such as *pcap* files, as a set R of elements ($srcip$, $desip$, $protocol$, $srcport$, $desport$, $time$, $payload$), where the tuple represents a network packet at time $t \in T$. Notice that the *payload* may or may not be empty. T_{train} and T_{test} are two-dimensional matrices, where each row is a network flow and and each column is a statistical feature of the selected feature set.

B. Model

Sparsity Observation: Bots belonging to the same botnet share the same group behavior in exchanging C&C messages since bots are hard-coded to perform some structurally repetitious tasks. Therefore, there exist n statistical features which group the flows of the same botnet into one or more separate clusters. We call the given n statistical features *effective* if the generated clusters of different botnets are distinctive from one another. More precisely, we assume that botnet clusters are sparse w.r.t. n effective statistical features. That is, for the majority of clusters, there is no close cluster in its neighborhood.

Similarity Estimation: The similarity (i.e., distance) between a flow $\mathbf{F}$ and a cluster $\mathcal{C}_{\mathbf{FP_j}}$ is measured with function Sim, which maps a pair of flows consisting of n statistical features to the interval [0 1]. For a flow $\mathbf{F} = (f_1, f_2, \ldots, f_n)$ and a cluster $\mathcal{C}_{\mathbf{FP_j}}$ with its center $\mathbf{FP_j} = (fp_1, fp_2, \ldots, fp_n)$, Sim, a $k-$similarity measure, is defined as

$$\mathsf{Sim}(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) = \cos \sum_{i=1}^{k} \left(\frac{f_{min_i(\mathbf{F},\mathbf{FP_j})} - fp_{j_{min_i(\mathbf{F},\mathbf{FP_j})}}}{max(\bar{f}_{min_i(\mathbf{F},\mathbf{FP_j})}) - min(\bar{f}_{min_i(\mathbf{F},\mathbf{FP_j})})} \right)^2 \quad (1)$$

where $min_i(\mathbf{F}, \mathbf{FP_j})$ returns the index of the ith closest distance feature pair among all features in $\mathbf{F}$ and $\mathbf{FP_j}$. $max(\bar{f}_k)$ and $min(\bar{f}_k)$ return maximum and minimum values of feature k over the observed flows in T_{train}, respectively. For example, given $\mathbf{F} = (0.2, 0.5, 0.4)$, $\mathbf{FP_j} = (0.2, 0.3, 0.3)$, $max(\bar{f}_1) = 0.4$, $max(\bar{f}_2) = 1$, $max(\bar{f}_3) = 0.8$, $min(\bar{f}_1) = 0$, $min(\bar{f}_2) = 0$, $min(\bar{f}_3) = 0.1$, then a 2-similarity measure, is $\mathsf{Sim}(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) = \cos((\frac{0.2-0.2}{0.4-0})^2 + (\frac{0.4-0.3}{0.8-0.1})^2) = 0.99$.

Bots randomize their behaviors. Since small values are less likely to be affected by evading methods, we measure the similarity solely based on top k minimum features. Our similarity function is robust against bots with such behavior (see Section V-D for details).

Nontrivial Feature Selection: From above, Sim measures the k closest feature pairs in $\mathbf{F}$ and $\mathcal{C}_{\mathbf{FP_j}}$. This can be explained as follows.

We assume that there exists an oracle $\mathcal{O}(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) \in [0\ 1]$ that when $\mathbf{F}$ and $\mathcal{C}_{\mathbf{FP_j}}$ are highly similar, it outputs 1. Let $T(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) = \{(f_{i_j}, fp_{i_j}) | 1 \le j \le m\}$ be a set of selected feature pairs by oracle $\mathcal{O}$ for measuring the similarity between $\mathbf{F}$ and $\mathcal{C}_{\mathbf{FP_j}}$. Sim uses a set of feature pairs $T_k(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}})$, which is chosen by sorting the distances of feature pairs in ascending order and picking top k closest feature pairs.

Since an ideal similarity function utilizes the most analogous feature pairs, it is crucial to evaluate its reliability. Instead of considering whether a feature pair $(f_l, fp_l) \in T_k(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}})$ is similar to a $(f_i, fp_i) \in T(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}})$, we want to evaluate how likely (f_l, fp_l) belongs to $T(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}})$ given a similarity function Sim. The probability that a collection of feature pairs is a true match given the measurement value z can be expressed as

$$\Pr[T_k(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) \subseteq T(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) | \mathsf{Sim}(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) = z] \quad (2)$$

Moreover, the probability of observing a measurement value z given a corresponding feature pairs is

$$\Pr[z | T_k(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) \subseteq T(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}})] \quad (3)$$

Then, the accuracy of the similarity function is

$$Acc(\mathsf{Sim}) = \int_0^1 \Pr[T_k(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) \subseteq T(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) | z] \times \Pr[z | T_k(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) \subseteq T(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}})]\, \mathrm{d}z \quad (4)$$

where z is uniform and $Acc(\mathsf{Sim})$ is the accuracy of k pairwise feature similarity measurement. Since

$$\Pr[T_k(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) \subseteq T(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) | z] \le 1 \quad (5)$$

$$\text{and} \int_0^1 \Pr[z | T_k(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) \subseteq T(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}})]\, \mathrm{d}z = 1$$

The accuracy of measurement is $Acc(\mathsf{Sim}) \le 1$ and $Acc(\mathsf{Sim}) \simeq 1$ if and only if $\Pr[T_k(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) \subseteq$

[1]Notice that the randomization process cannot be arbitrarily large. If bots transmit C&C commands within a very large interval, the utility will be lost.

[2]Throughout this paper, the terms cluster centroids and fingerprints are interchangeable.

$T(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}})|z] = 1, \forall \Pr[z|T_k(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) \subseteq T(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}})] > 0$. The remaining task is to show that the output of $\mathcal{O}$ and Sim are close. This requires that $T_k(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) \subseteq T(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}})$.

$$Acc(\mathsf{Sim}) \simeq Acc(\mathcal{O}) \iff T_k(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) \subseteq T(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) \quad (6)$$

We cannot formally argue that our feature selection technique ensures $T_k(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) \subseteq T(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}})$. That is, $T_k(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}})$ may contain a pair of feature $(f_l, fp_l) \notin T(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}})$ but still $Acc(\mathsf{Sim}) \simeq Acc(\mathcal{O})$. Sim uses the same feature space as $\mathcal{O}$ except that it uses k feature pairs with the closest distance among n feature pairs. Since Sim uses Euclidean distance to map the feature pairs to the interval [0 1], the choice of closest feature pairs increases the likelihood of $T_k(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) \subseteq T(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}})$. We conduct an experiment to verify the accuracy of nontrivial feature selection technique for detecting flows of Neris, Storm and Waledac botnets. The accuracy of detection algorithm only based on Sim function for Neris, Storm and Waledac botnets are 93.04%, 93.82% and 99.76%, respectively.

This implies that given a set of effective features and a $\mathbf{F}'$ perturbed instance of $\mathbf{F}$,

$$\Pr[|\mathsf{Sim}(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) - \mathsf{Sim}(\mathbf{F}', \mathcal{C}_{\mathbf{FP_j}})| < \epsilon] > 1 - \gamma$$

for a small γ. As a result, it is less likely for a bot to bypass the similarity function by randomizing its behavior, since it is not aware of which features will be used in Equation 1.

Re-identification: We frame the problem of detecting botnets as a re-identification problem, which is similar to the de-anonymization problem [6]. De-anonymization aims to learn unknown user ratings or attributes using a subset of known attributes.

There are some important differences between re-identification and de-anonymization algorithms. First, the feature values (equivalent to the ratings in the Netflix dataset) are variable and related to bots behaviors. This causes a significant challenge to direct application of traditional de-anonymization algorithms that deal with static features. Secondly, the features in this case are implicit, as it is derived from the activity of bots, whereas, in de-anonymization algorithms, the ratings are explicitly presented by users. Thirdly, in the traditional de-anonymization algorithms, a partial information about a user is given and the goal is to find more information by referring its similar users. In contrast, we have complete information about our targets. In addition, the sparsity of the test dataset can not be guaranteed. Therefore, traditional de-anonymization algorithms are not directly applicable in our setting.

We aim at re-identifying bots by obtaining some information from their network flows. What makes re-identification possible is that feature values are not given randomly, but according to complex behaviors of bots that are not known in advance. Some of the behaviors are intuitive while others need to be unveiled through proper analysis of the dataset.

Definition 1: A database Δ of clusters is $(\epsilon, \delta)-$sparse w.r.t. similarity function Sim, if most clusters do not have any similar cluster (or near neighbor).

$$\Pr_{\mathbf{FP}}[\exists \mathbf{FP}' \neq \mathbf{FP}, \mathsf{Sim}(\mathbf{FP}, \mathcal{C}_{\mathbf{FP}'}) \geq \epsilon] \leq \delta$$

Given several similar flows with a center $\mathbf{FP}$, Definition 1 shows that for the majority of flows, there isn't a single similar flow in another cluster $\mathcal{C}_{\mathbf{FP}'}$ with an overwhelming probability. In another word, the majority of the bot flows are very similar to their fingerprint, i.e., share the same or very close attribute values.

Let $\bar{\Delta}$ denote the real fingerprints of P2P applications. The actual network flows pf botnets try to deviate from known fingerprints in order to evade detection. This is modeled by a probabilistic algorithm Aux, which on input $\mathbf{FP_j} \in \bar{\Delta}$, outputs aux, a perturbed or imprecise network flow set. We assume that the feature values of a randomly selected fingerprint $\mathbf{FP_j}$ and the corresponding feature values in the perturbed network flows aux are close, i.e., $\mathsf{Sim_i}(\mathbf{F} \in aux, \mathcal{C}_{\mathbf{FP_j}}) > 1 - \gamma$ for some small γ and $1 \leq i \leq k$, where $\mathsf{Sim_i}$ is a pairwise measurement between f_j and fp_j.

Definition 2: An arbitrary network flow $\mathbf{F} \in \mathsf{Aux}(\mathbf{FP_j} \in \bar{\Delta})$ can be (θ, ω)-identified w.r.t. similarity function Sim if there exist an algorithm $\mathcal{A}$ which, on inputs $\mathbf{F}$ and $\bar{\Delta}$, outputs

- $\mathbf{FP_j}$ s.t. $\Pr[\mathsf{Sim}(\mathbf{F}, \mathcal{C}_{\mathbf{FP_j}}) > \theta] > \omega$, if $\mathbf{FP_j} \in \bar{\Delta}$
- $\perp$ with probability at least ω, otherwise,

where the probability is taken over $\bar{\Delta}$ and the randomness of $\mathcal{A}$.

$\mathcal{A}$ starts with a database $\bar{\Delta}$ of fingerprints, and uses this to locate candidate fingerprints $\mathbf{FP_j} \in \bar{\Delta}$ which are close to an actual flows $\mathbf{F} \in T_{test}$.

In Definition 2, the re-identification algorithm $\mathcal{A}$ finds a candidate cluster which is similar to the target flow. The second condition boasts the performance of the re-identification algorithm $\mathcal{A}$ by detecting whether the target flow (bot) is a part of the fingerprint dataset, or has not been released at all.

Notice that, our T_{train} only includes some possible benign and botnet communications. Hence, $\Delta \subseteq \bar{\Delta}$ represents the database of fingerprints extracted from T_{train}. We may further assume that Δ contains some noise. That is, Δ approximates the centroids in the absence of actual centroids because the given T_{train} may have incomplete information regarding botnets. More precisely, assuming T_{train} contains some noise denoted by $Z_{T_{train}}$, and the noise generated by Aux algorithm is Z_{Aux}, then $\mathcal{A}$'s task is similar to the scenario where T_{train} contains complete information, and noise $Z_{T_{train}} + Z_{Aux}$ is generated by Aux. From the perspective of our re-identification algorithm, there is no difference between the perturbation of T_{train} dataset and inaccuracy of centroids in Δ. In either case, there is a small deviation between the feature values in an arbitrary network flow $\mathbf{F}$ and the same values in Δ. Our definition of re-identification allows $\mathcal{A}$ to identify its target flow as long as it is sufficiently similar w.r.t. Sim to a known fingerprint, which is used to label the target flow with high probability.

IV. System implementation

A. Feature extraction module

The performance of re-identification algorithm depends on a set of carefully selected features. We select robust features that are discriminable among bots as follows: APL (Average packet length), PV (Standard deviation of packet length), PX (Total number of packets exchanged), FPS (Length of the first packet), IOPR (Ratio between the number of incoming and outgoing packets), BS (Average bytes-per-second), PPS (Average packets-per-second), and Duration (Flow duration). As mentioned before, the bot communications have more regularity. Hence, the exchanged data among bots have less randomness. Our observation over the dataset shows that flows associated with legitimate P2P applications usually carry a larger amount of data. Consequently, legitimate P2P applications exchange more network packets which are captured in features *PX, IOPR, BS, PPS*. Long duration flows that carry a low traffic volume have more likelihood of being a botnet activity. Therefore, *Duration* together with packet length features can capture such a phenomenon.

B. Clustering botnet flow

In this section, we obtain a set of fingerprints Δ^j for each botnet type j.

We observe that the bots belong to the same botnet type share similar patterns in their P2P communications. We use the *Affinity Propagation* (AP) clustering method [9], which does not need to determine or estimate the number of clusters before running the algorithm. We invoke (AP) clustering over all infected hosts that run botnet type j, and collect the outputs into a set of clusters $\{\mathcal{C}_1^j, \mathcal{C}_2^j, \ldots, \mathcal{C}_p^j\}$. We define the corresponding set of fingerprints for botnet type j as $\Delta^j = \{\mathbf{FP_1^j}, \mathbf{FP_2^j}, \ldots, \mathbf{FP_p^j}\}$.

For a flow $\mathbf{F}$ of a host that runs botnet j, the probability of finding an appropriate fingerprint $\mathbf{FP_k} \in \Delta^j$ is quite high,

$$\Pr[\exists \mathbf{FP_k^j} \in \Delta^j, \mathsf{Sim}(\mathbf{F}, \mathbf{FP_k^j}) \leq d_j\%] \geq 1 - \gamma \qquad (7)$$

for a small γ and a predefined distance d_j. Therefore, the similarities of fingerprints for different hosts indicates how well a fingerprint is suited to a cluster center. Moreover, we obtain some statistical information such as minimum and maximum values of a feature as required in Equation 1. Finally, we obtain the fingerprint database Δ of the botnets and benign applications in T_{train} as $\Delta = \bigcup_j \Delta^j \cup \Delta'$, where Δ' is the fingerprints of benign applications.

C. Training Model

Training Model takes the fingerprint database Δ and T_{train} as input. For each application type j, it outputs $\langle d_j, k_j, CS_j \rangle$, where the pair (d_j, k_j) is a customized parameter for Sim function to tag a flow. We also train a one-class classifier CS_j for each application type j to give a score over a time interval W_i.

For each application type j, we find a pair (d_j, k_j) to tag a flow. Recall that Sim uses k closest distance feature pairs for detecting different P2P applications. We need to obtain this k for each P2P application. Moreover, for each P2P application type j, a threshold $\mathsf{Sim}(.,.) \geq d_j\%$ is set during the training phase to restrict the distance between a flow and its corresponding closest fingerprint.

We use Algorithm 1 to obtain a pair (d_j, k_j) for each P2P application j. Algorithm 1 has as input Δ, ω, T_{train}, and $P2PApps$ (the set of types of applications in T_{train}). It begins by collecting network flows of all hosts in T_{train} that run application type j. Next, it chops the flows into time intervals W_i of size ω for all $i \in \{1, \ldots, n\}$. It tries different d_j and k_j to find a $\mathbf{FP} \in \Delta^j$ s.t. $\mathsf{Sim}(\mathbf{F}, \mathcal{C}_{\mathbf{FP}}) \geq d_j\%$ by using $k = k_j$. Finally, it chooses customized pair (d_j, k_j) for each P2P application for maximizing its accuracy.

Algorithm 1: Obtaining bot detection parameters

```
Data: $T_{train}$, $\Delta$, $\omega$, $P2PApps$
1 For each  app $j \in$ P2PApps
2  For each $d_j \in \{1\%, \ldots, 50\%\}$ and $k_j \in \{2, \ldots, 8\}$
3     For each $\mathbf{F} \in W_i$ from $H_l$ running app $j$ in $T_{train}$
4        $\mathbf{F}.dist \leftarrow max\{\mathsf{Sim}(\mathsf{F}, \mathcal{C}_{\mathbf{FP}})$ with $k = k_j\}_{\forall \mathbf{FP} \in \Delta^j}$
5     $counter[k_j][d_j]$ = (# Of correctly labeled flows $\mathbf{F}$ with
6                       $\mathbf{F}.dist = d_j$ and $k_j$ feature pairs in Sim)
7  Output $k_j, d_j = argmax(counter[k_j][d_j])$
```

A flow $\mathbf{F}$ is tagged as running application type j if $\mathsf{Sim}(\mathsf{F}, \mathcal{C}_{\mathbf{FP}}) \geq d_j\%$ with $k = k_j$, for some $\mathcal{C}_{\mathbf{FP}} \in \Delta^j$.

Figure 1a shows the accuracy ratios of Storm botnet as a function of d and k, where we fix the time interval window size $\omega = 15$ minutes. The best accuracy is obtained when $k = 3$ and $d = 25\%$. We obtain $k = 3$ and $d = 8\%$ for Neris botnet and $k = 7$ and $d = 41\%$ for Waledac botnet, etc.

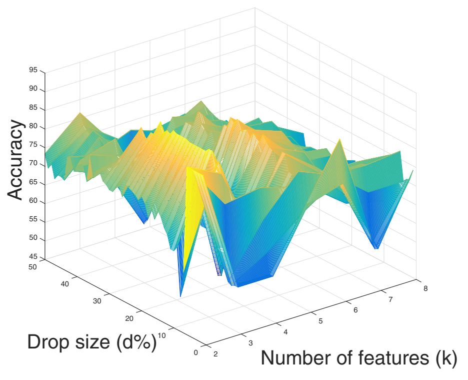
Figure 1b shows the accuracy ratios as a function of the time interval window size ω, where we fix d and k. The accuracy ratio improves notably when ω increases from 10 to 15 minutes. Then it decreases when ω is more than 15 minutes. Therefore, we adopt ω to be 15 minutes.

At the end of each time interval W_i of all hosts running application type j, we create a dataset $\mathcal{TD}_j$ as follows. We invoke Sim function over each flow $\mathbf{F}$ in time interval W_i to find the best matching cluster. Next, we group the results based on the types of matching clusters as $instance(W_i) = \langle app_1, count_1 \rangle, \langle app_2, count_2 \rangle, \ldots, \langle app_n, count_n \rangle$, where $\langle app_x, count_x \rangle$ denotes the number of flows in W_i that are tagged as application type x. We repeat this task over all time intervals and let $\mathcal{TD}_j = \{(instance(W_i), j) | i = 1, \ldots, n\}$. We further add some instances of other application $b \in P2PApps$ (where $b \neq j$) into $\mathcal{TD}_j$ to balance the dataset.

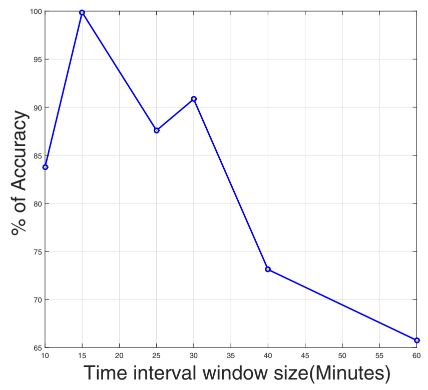
Finally, we train a customized classifier for each P2P application in $P2PApps$. For each $\mathcal{TD}_j$, we build and output the corresponding classifier CS_j.

D. Reduce false positive ratio in a host

In practice, not necessarily all P2P applications in a host will be active at the same time. Moreover, a host may run more than one active P2P application within a time interval W_i. In such a case, the host will be detected as a benign P2P host due to running a known P2P application, even though the host indeed hosts a P2P bot. For example, P2P bots may be involved



(a)



(b)

Fig. 1. (a) Accuracy ratio as a function of distance d and number of features k for Storm botnet. (b) Accuracy ratio as a function of ω.

in sending large amount of spam emails or downloading an update for the botnet software. This makes their network behavior closer to a benign P2P application. Consequently, classifying the P2P host requires further analysis.

We reduce the false positive ratio by applying majority vote over some consecutive intervals. We invoke the detection algorithm over 4 consecutive time intervals W_i, W_{i+1}, W_{i+2}, W_{i+4}. Let ID_i denote the intermediate decisions of the detection algorithm. We make the final decision using majority vote over ID_i, ID_{i+1}, ID_{i+2}, ID_{i+4}. The false positive ratio of Waledac botnet reduces from 1.39% to 0%. Similarly, Storm botnet experiences a drop from 0.67% to 0.11%.

We define $\mathbf{FDR}$ as a threshold vector that for each application type j shows how many intermediate decisions are required to minimize the FPR. In our experiment, for all applications in $P2PApps$, the highest FPR is achieved when using 4 time intervals. Therefore, we let $FDR_j = 4$ for all applications.

V. Evaluation

A. Dataset

Two datasets are involved in the creation and evaluation of our algorithm. The first dataset is the one used in the experimental evaluation of PeerRush [2]. It contains botnet traffic of Neris (3 hours, 10 hosts), Storm (24 hours, 13 hosts), and Waledac (21 hours, 3 hosts). The second dataset is captured from 179 hosts of our university that consists of benign traffic of type HTTP/S, FTP, P2P (10 hours, 179 hosts).

T_{train} is a subset of both datasets. We use the rest of the datasets as T_{test} including 371 botnet and 1779 benign samples.

B. Detection Algorithm

We begin by describing an algorithm template to identify the malicious hosts. The inputs are T_{test}, Δ, $\mathbf{d}, \mathbf{k}, \mathbf{CS}$, and $\mathbf{FPR}$. The output is a tag for each host indicating whether it is malicious or benign.

The main components of the algorithm are the scoring function, matching criterion and tagging algorithm. The scoring function associates a numerical value s to each flow $\mathbf{F} \in T_{test}$ based on how properly it suits to a fingerprint in Δ. The matching criterion is invoked every ω minutes over the results of the scoring function and outputs an intermediate decision. Finally, the tagging algorithm makes the final decision after observing some consecutive intermediate results.

Detection Algorithm: The following is a simple instantiation of the above template.

- For each flow $\mathbf{F}$ of host H_l in T_{test}, $\mathcal{A}(\Delta, \mathbf{F}, \mathbf{d}, \mathbf{k}) \xleftarrow{j} typeof(max\{\mathsf{Sim}(\mathbf{F}, \mathcal{C}_{\mathbf{FP}})\}_{\forall \mathbf{FP} \in \Delta})$. The score of a flow $\mathbf{F}$ is determined by its closest fingerprint $\mathbf{FP} \in \Delta$ s.t. $\mathsf{Sim}(\mathsf{F}, \mathcal{C}_{\mathbf{FP}}) \geq d_j\%$. If the set is empty, output $\perp$. Otherwise, return the type of fingerprint $\mathbf{FP}$.
- For each W_i, the matching criterion invokes $\mathcal{A}$ over all $\mathbf{F} \in W_i$ of host H_l. It generates an $instance(W_i)$ and feeds it to the previously trained classifiers $\mathbf{CS}$. It outputs the type of the classifier with maximum score as an intermediate decision $ID_i = typeof(W_i)$, which indicates that the traffic in W_i likely belongs to app ID_i.
- For each host $H_l, Tag(ID_i, ID_{i+1}, ID_{i+2}, ID_{i+4}, \mathbf{FDR})$ tags a host as malicious or benign after observing four consecutive intermediate decisions. It indicates that the given host runs application type j if $\sum_{ID_i=j} 1 \geq FDR_j$, where FDR_j is a trained threshold for application type j in the training phase.

Notice that there are three ways in which our detection algorithm may fail to tag a host correctly. First, an incorrect fingerprint is associated with the highest score. Second, more than one classifier outputs score s, where s is the maximum score. Third, the count of intermediate decisions are less than the predefined thresholds $\mathbf{FDR}$.

C. Accuracy

Table I (upper part) shows the detailed results for detecting Neris, Storm and Waledac botnets. We have more than 99% accuracy in detecting Neris and Storm botnets, and 100% accuracy in detecting Waledac botnet, with very low false positive ratios.

Our average TPR for detecting Storm and Waledac botnets is 99.14%, whereas the average TPR of PeerRush [2] is

TABLE I
THE DETAIL & OVERALL RESULTS OF OUR DETECTION ALGORITHM

Botnet	TPR	FPR	ACC	Precision	F1
Neris	100	0.23	99.78	81.82	90
Storm	98.29	0.11	99.66	99.31	98.80
Waledac	100	0	100	100	100
	TPR	FPR	ACC	Precision	F1
[3]	89.38	7.76	90.36	93.78	91.53
[4]	96.76	6.74	95.24	94.46	99.56
[5]	75.60	0.52	98.33	87.70	81.20
Ours	98.65	0.21	99.73	96.06	97.34

77.07%. Moreover, our average FPR is 0.05%, which is 7.5 times less than that of PeerRush algorithm. This shows the ability of our detection algorithm in producing more reliable results.

We implemented a number of other methods in order to conduct a comprehensive comparison using our dataset. Table I (lower part) summarizes the accuracy of Monsieur Poirot against the results of [3], [4], [5]. Monsieur Poirot outperforms the other algorithms by providing higher TPR, ACC and precision. Moreover, our detection algorithm can effectively detect P2P bots with a very low false positive rate.

D. Robustness

To test robustness of our detection algorithm against evading methods, we perform experiments by artificially adding noises to the network traffic according to some normal distributions. Since we do not have access to the source codes of tested botnets, we assume that a botmaster inserts arbitrary delay throughout communication traffic. Random delays are inserted between two subsequent packets by some normal distribution. We also simulate the scenario in which a botmaster changes packet lengths. This causes a change into APL, PV, FPS and BS features up to a defined mean μ and standard deviation σ. More precisely, we instantiate the probabilistic algorithm Aux as

$$\mathsf{Aux}(\mathbf{F} \in T_{test}) = (f_1 \pm \mathcal{N}(\mu, \sigma), \ldots, f_n \pm \mathcal{N}(\mu, \sigma)) \quad (8)$$

where μ and σ caps the level of perturbation on network flows. After inserting noises, we invoke feature extraction module (Section IV-A) to recompute new feature values.

Overall, the added noises to the *pcap* files decrease the accuracy. This implies that the choice of μ and σ introduces discrepancy between the original flow and the corresponding perturbed flow, i.e., $\mathsf{Sim_i}(\mathsf{Aux}(\mathbf{F}), \mathcal{C}_{\mathbf{FP_j}}) \not> 1 - \gamma$ for $1 \leq i \leq k$, when $\mu > 0$.

The delay inserted between communication packets has limited effect on the TPR. The overall loss is acceptable. For instance, noise added with $\mathcal{N}(6, 1)$ causes the TPR a decrease from 98.29% to 92.78%. We observe that if a Storm bot connects to its botmaster for longer than a time interval W_i, the probability of correctly tagging increases. Therefore, even when we increase the delay between communication packets, we obtain satisfactory results.

The noise added to packet lengths has more negative impact on TPR. For example, noise added by $\mathcal{N}(20, 3)$ causes TPR to decrease from 98.29% to 96.90%. For $\mathcal{N}(40, 10)$, we experience a drop from 98.29% to 68.04%. We found that increasing packet lengths cause our re-identification algorithm $\mathcal{A}$ to associate an incorrect fingerprint with a higher probability. In a sense, increasing packet length increases the probability of a malicious flow to be judged as a benign one. This indicates that the noise-tolerance of our feature set is restricted over large variation of packet lengths.

VI. CONCLUSION

In this paper we presented Monsieur Poirot, a novel re-identification approach for detecting botnets. We formulate the problem as an implicit de-anonymization problem. We also use a nontrivial feature selection technique to discover the best features for comparing two flows. The experimental results demonstrate its effectiveness for detecting malicious traffic.

The presented result is limited to detect three botnets. As a part of future plan, we aim to extend the capability of Monsieur Poirot to detect other botnets.

REFERENCES

[1] W. Wang and T. E. Daniels, "A graph based approach toward network forensics analysis," *ACM Trans. Inf. Syst. Secur.*, vol. 12, no. 1, pp. 4:1–4:33, Oct. 2008.

[2] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li, "Peerrush: Mining for unwanted p2p traffic," in *Proceedings of the 10th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, ser. DIMVA'13. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 62–82.

[3] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *Proceedings of the 2014 IEEE Conference on Communications and Network Security (CNS)*, ser. CNS'14. IEEE, 2014, pp. 247–255.

[4] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian, "Detecting p2p botnets through network behavior analysis and machine learning," in *Proceedings of the 9th International Conference on Privacy, Security and Trust (PST)*, ser. PST'11. IEEE, 2011, pp. 174–180.

[5] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals," *Comput. Secur.*, vol. 39, pp. 2–16, Nov. 2013.

[6] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, ser. SP '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 111–125.

[7] J. Zhang, R. Perdisci, W. Lee, X. Luo, and U. Sarfraz, "Building a scalable system for stealthy p2p-botnet detection," *IEEE transactions on information forensics and security*, vol. 9, no. 1, pp. 27–38, 2014.

[8] D. Dittrich and S. Dietrich, "P2p as botnet command and control: a deeper insight," in *3rd International Conference on Malicious and Unwanted Software, 2008. MALWARE 2008.* IEEE, 2008, pp. 41–48.

[9] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *science*, vol. 315, no. 5814, pp. 972–976, 2007.