P2P Botnet Detection using Behavior Clustering & Statistical Tests

Su Chang
Iowa State University
Dept. of Electrical and Computer Engineering
Ames, Iowa, USA
changs@iastate.edu

Thomas E. Daniels
Iowa State University
Dept. of Electrical and Computer Engineering
Ames, Iowa, USA
daniels@iastate.edu

ABSTRACT

Botnets are widely believed to be the most serious danger to the Internet. Most recent research on botnet detection focuses on centralized botnets and primarily relies on two assumptions: prior knowledge of potential C&C channels and capability of monitoring them. However, when botnets switch to a P2P (peer-to-peer) structure and utilize multiple protocols for C&C, the above assumptions no longer hold. Consequently, the detection of P2P botnets is more difficult. In this paper, we relax the above two assumptions and focus on C&C channel detection for P2P botnets that use multiple protocols (randomly chosen) for C&C. We first consider a clustering based node behavior profiling approach to capture the node behavior clusters in a network; we then propose two detection schemes using formal statistical tests on popular behavior clusters in this network. In brief, we detect C&C behavior by measuring its impact on one or more normal behavior clusters in a statistical way. In the initial evaluations, we validate the assumptions made in this paper under different real user traces from enterprise network environments. We then evaluate the proposed approaches to detect the C&C channel in both simple and realistic cases and achieve encouraging results in terms of high detection and low false positive rates.

Categories and Subject Descriptors

C.2 [**Computer**]: Communication Networks—*Security and protection*; I.2.6 [**Artificial Intelligence**]: Learning; I.5.2 [**Design Methodology**]: Pattern analysis

General Terms

Security, Algorithms, Experimentation

Keywords

Machine learning, anomaly detection, network security

AISec'09, November 9, 2009, Chicago, Illinois, USA.

1. INTRODUCTION

A bot is common parlance on the Internet for a software program that is a software agent [1]. According to [2], the common malicious use of botnets can be briefly classified as Distributed Denial-of-Service (DDoS) attacks, spamming, traffic sniffing, keylogging, spreading new malware, click fraud and mass identity theft. Previously, DDoS and spamming were usually highly concerned. However, more and more applications like keylogging, click fraud are being used for profit purposes. For example, our analysis on Agobot, Sdbot and Q8bot suggests one important characteristic of botnets: CDkey (e.g. Half-Life CDKey, CSKEY) grabbing. In addition, there is usually more than one botmaster controlling the botnet, which increases the botnet usage and makes it harder for traceback (our analysis shows one version of Sdbot allows up to four botmasters to control the IRC channel simultaneously).

Most current research on botnets focuses on C&C detection for centralized botnets, usually under two inherent assumptions: prior knowledge of potential C&C channels (e.g. IRC or HTTP) and capability of monitoring them. However, when botnets switch to more advanced ones (e.g. Nugache bot using random ports in C&C [13]), those assumptions no longer hold, making most current detection schemes unable to detect the more advanced P2P botnets.

1.1 Overview of the Proposed Approach

In this paper, we propose schemes to detect C&C channel of P2P botnets based on node behavior profiles proposed in [6], which characterizes the node behavior by jointly considering spatial and temporal correlations. The approach in [6] takes into account the fact that network applications are of different popularities and two different nodes may share similar behaviors, such as the behavior at time 1 of host i and the behavior at time 2 of host j in Fig. 1.

In general, from the studies of P2P botnets in [33] [35] [7] [31], the advantages (e.g. robustness) of P2P botnets is mainly from the structure rather than the specific P2P protocols. Consequently, many P2P botnets [33] [35] [7] [31] proposed in the literature can randomly choose from either P2P protocols and/or non-P2P protocols for C&C. In this paper, given the trace data we have, we only consider the latter one and regard the detection of P2P botnets using P2P protocols as our further work. Utilizing multiple non-P2P ports in C&C for P2P botnets is easy to implement and possibly widely used in current botnets. For example, once the Nugache bot was altered to use random high-numbered service ports, it dropped off of nearly everyone's radar [13].

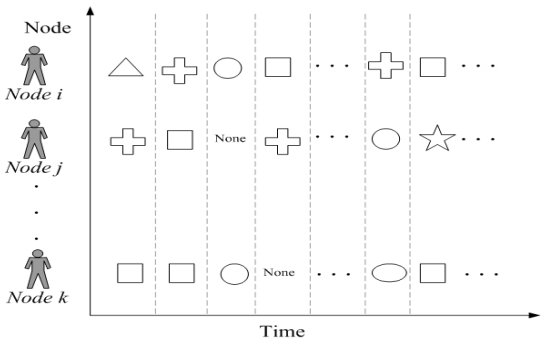

Figure 1: Examples of node behavior overlap

Therefore, our focus is to design detection schemes without previous two assumptions. Given the unique characteristics (small volume, short lifetime and mixed with normal traffic) of C&C behavior, and the correlation of node behaviors, we are able to design anomaly detection schemes based on formal statistical tests to determine if there are notable underlying botnet C&C behaviors in the network. The main idea of our approaches is that the botnet C&C behaviors will cause statistical changes for the existing behavior clusters, especially popular ones. To be specific, we make the following contributions:

1. Capture the normal traffic using node behavior clusters, formulate the problem of detecting C&C traffic as detecting additional behaviors over the existing normal behavior clusters.

2. Propose two anomaly detection schemes using statistical tests on popular behavior clusters, under the assumption that certain metrics of popular behavior clusters will be statistically stable.

3. Validate the assumptions made in this paper using real traces from enterprise networks.

4. Evaluate the detection schemes in both simple and realistic scenarios.

In the initial evaluations, similar to [15], since the enterprise network is less studied than the Internet [23], we consider the enterprise network from [23] in this paper. Moreover, in addition to simply combining two independent traffic types (bot traffic and normal traffic) during the evaluation, we consider the case where the bot traffic is correlated with the normal traffic. That is, to avoid being detected, the botmaster can randomly assign the C&C ports and peers within the same range of the targeting network to each bot. This correlation is usually not considered in the current research, but we believe this case is more realistic and could be adopted by the botmaster easily.

The BotMiner in [15] is also designed for P2P botnet detection. It did a good job of detecting the C&C channel of P2P botnets under the assumption of frequent communication between peers and capability of detecting attacks launched by bots after C&C. However, as four histograms in C&C communications are needed during detection, it mainly focuses on detecting P2P botnets requiring frequent communication between peers (e.g. PULL based). As botnets do not require frequent communications among peers in general (except the bot located in the network using NAT), it is uncertain that their scheme can detect general P2P botnets (e.g. the sample needed for histogram may not be large enough for general botnets). Moreover, detection of attacks generated by bot is a necessary condition in [15] to detect the C&C channel. Given many attacks used by botnets (e.g. clickfraud) are much harder to detect, it is uncertain that their scheme can detect C&C channel of P2P botnets launching more sophisticated attacks. Therefore, more work on detecting C&C of P2P botnets needs to be done. In this paper, our work aims to detect C&C channel without the above constraints, and only rely on the statistical nature of the node behavior clusters.

This paper is organized as follows: We discuss the related work in Section 2. In Section 3, we introduce the correlation based node behavior profiling approach, and two anomaly detection schemes. In Section 4, we first validate the assumptions made for detection, and evaluate the proposed schemes by using P2P botnet C&C traffic with the real trace from enterprise network environments, both simple and realistic cases are further considered. We conclude the paper in Section 5.

2. RELATED WORK

The structure of P2P botnets has been widely discussed in the literature. In [18], the authors list the timeline of captured P2P botnets. The authors in [10] did a thorough study on botnet structures and show that random botnets are highly resistant to different removal methods. Several different P2P botnets have also been proposed such as hybridbot [33], randombot [35], lcbot [7] and superbot [31]. To be specific, as illustrated in Fig. 2 (each bot has a peerlist of 5), each bot in randombot/hybridbot maintains a peerlist for communication during C&C; superbot divides the botnet into multiple groups, with each group having a supernode (middle node in Fig. 2), the C&C among supernodes is like randombot, while the C&C within each group is like centralized botnets; lcbot is a mixture of superbot and randombot, each bot has a peerlist with one peer outside its group and others within its group, it has a better performance in terms of mean coverage after certain removals and low overhead during C&C as shown in [7]. PUSH or PULL based communications [15] can be further used for each kind of P2P botnet, where "PUSH" refers to relaying commands to its peers during C&C and "PULL" refers to checking peers periodically for commands. It is also worth pointing out that, in all these P2P botnets, there is no limitation to use any specific protocols for C&C, thus different protocols can be used for C&C, (e.g. there are reports of botnets using gmail and skype for C&C). In practice, it is easy to implement multiple protocols for C&C in P2P botnets, and it is very likely to happen widely nowadays (e.g. Nugache bot using random ports for each bot during C&C [13]).

Many schemes have been proposed in the literature to detect botnets with a centralized structure. To summarize, those schemes are based on one or more of the following techniques: honeypot/honeynet [2], DNS inspection [35] [24], DNSBL inspection [26], traffic pattern recognition/clustering [4], temporal or spatial correlation [15] [17] [16], and many other schemes [8] [14] [19] [32] [29]. We only discuss the recent advances in botnet detection. To be specific, the authors in [26] proposed a detection scheme on

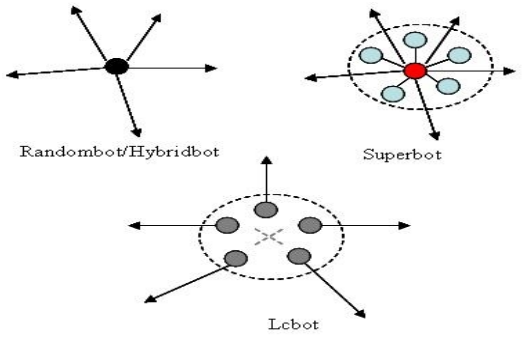

Figure 2: P2P botnet structures

DNSBL counter-intelligence. However, it is only focused on finding botnet members generating spams. In [27], by aggregating traffic of similar payload, same external destination, and internal hosts with similar OS platforms, the authors proposed a system called TAMD to detect malware including botnets. In [19], machine learning is used to identify the bot controllers of IRC based botnets. The authors in [15] [17] [16] proposed three kinds of detection approaches: BotHunter, BotMiner, and BotSniffer. BotHunter detects the bots by associating IDS events to a user-defined bot infection dialog model, and it is a passive detection system. BotSniffer is designed to detect centralized botnets by using horizontal correlation. BotMiner also utilizes a horizontal correlation approach that examines correlation across multiple hosts [15]. It did a good job of detecting the C&C channel of P2P botnets under the assumption of frequent communication between peers and detection of attacks launched by bots after C&C.

Most schemes rely on the two assumptions for detection or are based on the detection of worms. Unfortunately, they become ineffective when botnets shift to other structures, other propagation methods or use encryption in C&C. Moreover, many new features of botnets discussed in the recent literature make the existing countermeasures less effective. In addition to P2P structure, encryption, use of common protocols for C&C are also within the main directions of the evolution. Encryption makes it more secure for a botmaster to control the botnet, resulting in the inefficacy of schemes based on signatures or anomaly detections using character distribution (e.g. n-gram distribution). Normally, symmetric encryption is expected, but it is possible to make use of the PKI structure [3]. C&C by other commonly used protocols makes the communication among bots more covert [13]. Consequently, there are reports of botnets using Skype [21] and Gmail [5] in C&C. It is also possible that a botnet could use multiple protocols for one C&C cycle [13], making detection even more difficult. Other evolutions are listed in Table. 1. However, little research has been done towards these directions.

3. NODE BEHAVIOR BASED DETECTION

In this section, we first discuss the attributes used for profiling. Clustering is then used to find the representative behavior clusters. Statistical tests are further discussed to construct the proposed detection algorithms.

3.1 Attribute Selection & Behavior Clustering

According to [6], the attributes are chosen from the observation of the action sequence of normal users:

1. A normal user first decides which application/service he wants to use (e.g. in order to do some searching, HTTP is used).

2. Among those destinations providing the wanted service, he then determines which destination(s) he is interested in, this usually refers to the contents provided by the destination (e.g. To do a search, the user could use Yahoo or Google, and the user may choose Google).

3. Traffic is further generated when he visits the chosen destination.

The action sequence differs greatly between the normal user and the botnet. Since the botnet is dynamic: peers in the botnet can be dynamically shut down or removed from the botnet at any time, a bot may first generate traffic to find the online peers on certain ports from its peer list, and then send a command to all the available peers. On the other hand, it is very unlikely that a normal user (or a majority of normal users) generates the normal behavior this way. Therefore, this difference provides inherent guidelines in selecting and simplifying the attributes for node profiling.

Consequently, for attributes with large dimensions, e.g. the dimension of destinations may be quite large, we need to simplify the attributes in a way more favorable for normal behaviors while unfavorable for C&C. Firstly, we categorize the destinations as least, less, moderate and most popular categories, and only consider the corresponding packets generated in each category (represented as $pkt_{d,i,t}$). This is because although normal users are capable of choosing arbitrary destinations, they usually associate themselves on a small range of destinations of different popularity (or in-degree, a destination in-degree is the number of nodes connecting to it). On the other hand, the peers chosen in P2P botnets are random regardless of the destination popularity. The four categories are achieved by sorting the in-degree of destinations, dividing the in-degree into four quartiles and counting the packets in each quartile for each node.

Secondly, we consider all the services as attributes for node profiling, as the combination of all the normal services should also be considered normal. In addition, we only consider TCP traffic, as most of C&C traffic uses TCP (it is similar to design behavior profiles under UDP). Within each application, we consider the packets sent for that application. This is because the node behavior can be better characterized by the packets sent rather than the bytes sent, as bytes only represent the accumulated level of packets. Instead, we consider the total amount of traffic (in bytes) generated and received using tg_i and tr_i for node i. On the other hand, we do not consider the information such as inter-arrival time for each service, because it is very sensitive to network conditions, such as congestion level, routing policy, buffer management and location of monitors. Therefore, it may not be a good indicator of node behaviors. Thus the node behavior $x_{i,t}$ for node i at time t is $x_{i,t} = \{(pkt_{i,j,t}, tg_{i,t}, tr_{i,t}, pkt_{d,i,t}) | 1 \leq i \leq N, 1 \leq j \leq M, 1 \leq t \leq T, 1 \leq d \leq 4\}$ [6].

As there is no prior knowledge on how many common behavior clusters are shared by different nodes in the net-

Table 1: Evolution of bot techniques

	Traditional	Evolution
Botnet size	Large [9]	Small [9]
Bot bandwidth	Varies [9], usually small	Large [9]
C&C	Centralized (IRC)	Fully or partially distributed [10] [35] [33] [31]
Authentication	Plain text	MD5 [11], PKI [33]
C&C channel	IRC protocol	Various protocols, e.g. HTTP [24], Voip [21], gmail [5], random ports [13]
Communication	Plain text	SSL [5], encryption, information hiding or covert channel
Disguise IP	No	Yes, fake BGP route announcement [25]
Propagation	Pure worm-like	Propagate by command two-phase [35], superbot [31]
Systems	MS Windows	Windows Linux, Mac, etc.
Others	Kill Anti-Virus	Kill Anti-Virus Honetypot aware [35], NAT enabled [35], mimic human responding time interval more intelligent DNSBL lookup

work, agglomerative clustering is used to find possible clusters. Since the data set under consideration is clean [23], we first identify the active nodes that initiate connections, as those nodes are the sources of most, if not all, network events. This is achieved by using the method described in [34], which is based on the SYN packet. At the same time, the destinations' in-degrees are calculates and sorted. Then, at a fixed time interval, we calculate the values of the attributes for each active node, which is $x_{i,t}$ in the above discussions. We also remove the $x_{i,t}$s with $tr_{i,t}$ equal to 0 to remove the silent nodes. Finally, we compute the log values of those attributes. The classic agglomerative clustering algorithm is further used [6], it treats each $x_{i,t}$ as a cluster at the beginning, then calculated the pairwise distance for any two points (or $x_{i,t}$s), and combined two points or clusters into one cluster if the distance between them is below a threshold T_h. The criterion to compare the distance on combining clusters is the largest distance, and we set $T_h = 0.25$ in the evaluation section (for the traces we have analyzed, the proposed schemes achieve similar performances when T_h is chosen from 0.2 to 0.3). The distance is measured by the extended Jaccard distance which is defined as [30]: $d(x_1, x_2) = 1 - \frac{x_1' x_2}{||x_1||^2 + ||x_2||^2 - x_1' x_2}$.

3.2 Behavior based Detection Algorithms

Of all the behavior clusters identified in the training data, we are interested in those which are most common or popular. This is because we can get enough samples from those popular behavior clusters, and have more accurate estimations on the metrics of interest. Therefore, in terms of detection, if there is any group behavior in the new sample data which is unseen in the training data, it will either cause the proportion change of popular behavior clusters (e.g. the original behavior i may switch to another behavior j, affected by the unseen behavior in the new data); or the intra-cluster distance change within each popular behavior cluster (if the affected behavior still belongs to cluster i). Generally, both these cases will happen simultaneously in one or more behavior clusters, so we can design algorithms based on different statistical tests for each case. Consequently, the assumption made in the proposed detection approaches is that the metrics for popular behavior clusters used for detection should be statistically stable across training data and new data (we validate this assumption in the next section).

3.2.1 Behavior Proportion based Test (BPT)

For the l popular behavior clusters identified (in general, we consider the first l clusters, e.g. $l = 3$), consider the ith popular behavior ($i \in l$) a specified property in the new data, statistical test on population proportion of specified property can be applied in testing if the proportion of the ith behavior in the new sample is from the same population as the training data. To be specific, given the training data sufficiently large, the estimate of proportion of the popular behavior i can be considered as an accurate estimate for the corresponding population. For any new sample captured, if it is from the same population as the training data, under a predetermined significance level α, the test will give no evidence of rejecting the non-hypothesis. On the other hand, if they are from different populations, the test will reject the non-hypothesis.

In detail, the new sample size n and the number of occurrences of popular behavior i (denoted by Num_i, Num_i is then a random variable) may vary from time to time. At time interval t, two possible cases have to be considered, one is called the large sample case, with both $np_{i,0} \geq 10$ and $n(1 - p_{i,0}) \geq 10$, where $p_{i,0}$ is the true value of the proportion that popular behavior i in the population. The other is called the small sample case, where either $np_{i,0} < 10$ or $n(1 - p_{i,0}) < 10$, or both. Two kinds of tests are needed for these two cases. From [12], for the large sample test, both Num_i and the estimator $\hat{p}_{i,new} = Num_i/n$ are approximately normally distributed. Therefore, for the null hypothesis $H_0 : \hat{p}_{i,new} = p_{i,0}$ (with $H_a : p_{i,new} \neq p_{i,0}$), the test statistic

$$z = \frac{\hat{p}_{i,new} - p_{i,0}}{\sqrt{p_{i,0}(1 - p_{i,0})/n}} \quad (1)$$

The above null hypothesis will be rejected if $z \geq z_{\alpha/2}$ or $z \leq -z_{\alpha/2}$, otherwise, there is no evidence to reject the null hypothesis at significance level α.

For the small sample case [12], the small-sample test can be used, and it is based on binomial distribution. Usually in this test, a rejection region or acceptance region is used at given significance level α. We use the acceptance region in this paper, it is given by $[a, b]$, where

$$\alpha/2 \leq Bino(a; n, p_{i,0}) \quad (2)$$

$$Bino(b - 1; n, p_{i,0}) \leq 1 - \alpha/2 \quad (3)$$

If Num_i in the new sample does not belong to $[a, b]$, we reject the null hypothesis at significance level α.

3.2.2 Behavior Mean Distance based Test (BMDT)

Another metric of interest is the mean of intra-cluster distance of each popular behavior cluster. If there is a group

of subtle behavior added to the network, the mean value of intra-cluster distance may also change. Given the training data is large enough, we can accurately estimate the mean value for each behavior cluster. On the other hand, since the number of each popular behavior in any new sample is usually not large enough (< 30, as the time interval under consideration is short), to test if the mean values are statistically equal, one-sample t-test can be used [20]. So we have the null hypothesis $H_0 : d_{i,new} = d_{i,0}$ (with H_a:$d_{i,new} > d_{i,0}$), where $d_{i,new}$ is the mean intra-cluster distance of behavior cluster i in the new sample, and $d_{i,0}$ is the true value of the mean. The t statistic is given by [20] [12]:

$$t = \frac{d_{i,new} - d_{i,0}}{s/\sqrt{n}} \quad (4)$$

where $s = \sqrt{\frac{\sum y_j - d_{i,new}}{n-1}}$, and y_j the intra-cluster distance of member j in behavior cluster i. The degree of freedom of the t-test is $df = n - 1$.

3.2.3 Node Behavior based Detection Algorithm

Step 1: Training Data Processing
Identify first l popular behavior clusters and metrics of interest, e.g. proportion value, and mean intra-cluster distance from the training data.

Step 2: New Sample Data Processing
Remove the known attacks from the new sample, and then group each behavior in the new sample data to the closest behavior cluster by measuring the distance. Identify the first l popular behaviors in the new sample data. Calculate the metrics of interest.

Step 3: Behavior Proportion based Test
For each popular behavior i in the sample data, if it satisfies the large sample case, use equation (1) for the test, otherwise, use equation (2) and (3), at a given significance level α.

Step 4: Behavior Mean Distance based Test
Apply t-test using equation (4) for each popular behavior in the sample data, at the significance level α.

Step 5: Detection
For CTD: For each popular behavior i in the new sample data, if either test (or both) is rejected, call popular behavior i test positive; if r out of l such popular behaviors are test positive, generate an alarm, go to step 6. Otherwise, go to step 2.
For ITD: If there are two or more rejections in any test from l behaviors, generate an alarm, go to step 6. Otherwise, go to step 2.

Step 6: Indentification of C&C Nodes
For each behavior Y_i in the new data, find the closest behavior X_j in the training data, update Y_i by $Y_i = |Y_i - X_j|$. Use 2-means clustering to differentiate the C&C behaviors from normal behaviors, as normal behaviors will have $Y_i = |Y_i - X_j|$ around zero vector, while C&C behaviors will be centered at a vector point other than zero (this step is only designed for the common case discussed in next section). Go to step 2 for another new sample data.

Figure 3: Algorithm Description

With two kinds of formal statistical tests, it is possible to design several detecting algorithms. In this paper, we consider two of them. One is called independent test detection algorithm (ITD), which treats two tests (BMDT or BPT) independently, and generates an alarm when there are two or more rejections from any test in the l popular behaviors in the new sample data. The ITD algorithm is straightforward and simple. The next algorithm is called correlated test detection algorithm (CTD). It uses two parameters (r, l) and examines the first l popular behaviors. For each behavior cluster, if there is at least 1 rejection of tests by BPT or BMDT, the corresponding behavior cluster is said to be test positive. If at least r out of l clusters in the new sample are test positive, an alarm is generated, otherwise, there is no alarm. The reason of considering r behaviors together for detection in CTD is that, in practice it is impossible to set the α to be 0, that is to say, there are always a small number of false alarms; on the other hand, considering the subtle group behavior will affect more than one popular behaviors of the testing data, if there is no unseen group behavior, it would rarely happen that two or more popular behaviors do not pass the statistical tests at the same time interval (it also means that the false positive rates of CTD will be less than α). In addition, the main difference between two schemes is that CTD takes into account the proportion and intra-distance effects jointly. Consequently, we formulate the above detection algorithms in Fig. 3.

4. EVALUATIONS

We use the LBNL enterprise trace data which is the latest publicly available trace to evaluate the proposed approaches. In the LBNL trace collection [23], four NICs are switched periodically to capture traffic traces from 18-22 different subnets, with most traces being one hour long. We then choose six representative subnets, each of which is combined by two traces captured at different times and dates. We use time intervals of 10 minutes to profile the node behavior. In the evaluation part, we pick one time interval as the new data set, and the rest time intervals as the training data. Therefore, for each 2-hour trace, 12 possible tests are done in the evaluation. Specifically, we consider the trace data captured in port002, port003, port008, port010, port021 and port026 in this paper, with the size ranging from 100 to 1000 nodes.

4.1 Assumption Validation

To make the detection schemes effective, two assumptions have to hold. That is, proportion value and mean intra-cluster distance of the popular behaviors used in detection should be statistically stable for any single subnet.

For the proportion case, from Table 2, 16 (tests with value 1) out of 187 times ($16/187 = 0.086$) that the test wrongly rejects the null hypothesis, which is not far from the significance level at $\alpha = 0.05$ (since the number of nodes in port002 is very small, we only consider the first popular behavior). Especially, the first popular behaviors in port003, port008 and pot010 are not very stable for the proportion test. But other results do show certain stability of the proportion metric, so we believe the assumption on the proportion test can be considered valid at least for the LBNL trace. Similarly, from Table 3, 7 out of 187 times ($7/187 = 0.037$) that the test wrongly rejects the null hypothesis, this also matches the significance level at $\alpha = 0.05$ very well. In addition, the test on mean intra-cluster distance is more stable (as compared with proportion metric) across the traces under consideration. Therefore, we conclude that the assumptions are valid for the analysis in this paper.

Table 2: Proportion test results (1: reject; 0: not reject)

trace	clusters	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10	t=11	t=12
port002	(1)	0	0	0	0	0	0	0					
port003	(1,2,3)	(0,0,0)	(1,0,0)	(0,0,0)	(1,1,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(1,0,0)	(0,0,0)	(1,0,0)	(1,0,0)
port008	(1,2,3)	(0,0,0)	(1,0,0)	(0,0,0)	(1,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(1,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
port010	(1,2,3)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(1,0,0)	(1,0,0)	(0,0,0)	(0,0,0)	(1,0,0)	(0,0,0)	(0,0,0)
port021	(1,2,3)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
port026	(1,2,3)	(0,1,0)	(0,0,1)	(0,0,0)	(0,0,1)	(0,0,0)	(0,0,0)	(0,0,0)	(1,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)

Table 3: t-test results (1: reject; 0: not reject)

trace	clusters	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10	t=11	t=12
port002	1	0	0	0	0	0	0	0					
port003	(1,2,3)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,1)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
port008	(1,2,3)	(0,1,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,1)	(0,0,0)	(1,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
port010	(1,2,3)	(0,1,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
port021	(1,2,3)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
port026	(1,2,3)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(1,0,0)	(0,0,0)	(0,0,0)	(0,1,0)	(0,0,0)	(0,0,0)	(0,0,0)

4.2 False Positive Rate

Table 4 shows the false positive rates when different detection schemes are used. We set $l = 3$ and $r = 2$ for CTD and $l = 3$ for ITD. We can find that both schemes achieve low overall false positives rates (both less than 0.05). Especially, the CTD based detection reports only one false positive (port003 at interval 4) for the data we have analyzed. Further, if we remove the first behavior cluster in both port008 and port003 in the detection, there is no false alarm from both ITD and CTD schemes and the overall false positive rate drop to 0 for the traces we have analyzed. Therefore, from Table 4, we can find that although there are chances that the tests are wrongly rejected, by jointly taking into account the test results from different behavior clusters, we can achieve a lower false positive rate.

Table 4: False positive rates for each trace

trace	ITD(3)	ITD(2)	CTD(3, 2))	CTD(2, 2)
port003	8.3%	0	8.3%	0
port008	8.3%	0	0	0
port010	0	0	0	0
port021	0	0	0	0
port026	0	0	0	0
overall	3.3%	0	1.7%	0

4.3 Detecting the C&C in P2P Botnets

In this section, we consider two possible scenarios. One is called simple scenario where known botnet (Nugache bot) traffic [13] [28] is used for evaluation. This is achieved by adding the nodes generating Nugachebot traffic to the normal trace for evaluation. This is similar to the test used in the literature [15], where nodes using fixed port to generate bot traffic are independent with nodes generating normal traffic. Further, by noticing that any P2P botnet can randomly choose any port for C&C (especially those used in the training data), and the botmaster can avoid being detected by using those ports for C&C, we also consider this more realistic case in the evaluation. That is, in addition to generating the C&C traffic from a randomly chosen port, we enable each bot to randomly choose the port recorded in the training data for C&C, and only communicate with peers recorded in the training data to test the proposed schemes.

4.3.1 Detection in Simple Case

According to [28] [13] [22], the captured Nugache bot has a peerlist containing 22 peers and uses TCP port 8 for C&C. To evaluate the performance of the proposed detection schemes in the simple case, we use the last time interval as the testing data, and the earlier time intervals as the training data. For the testing data, in addition to the nodes generating the normal trace traffic, we add 100 Nugachebot nodes generating C&C traffic. To obtain the bot traffic, we run Nugache bot over VMware and capture the traffic on port 8. Since we also want to take into account PUSH based P2P botnets, we only consider one round of C&C (e.g. there is no frequent communication among peers). We use the detection rate for the evaluation; the detection rate is defined as the number of bots identified over the total number of bots in the data. In CTD algorithm, we set $l = 3$ and $r = 2$. Table 5 shows the results of detection rate under trace port010, port021 and port026. We can find, for all these traces, both schemes achieve 100% detection rate, we also use other traces from LBNL for the evaluation and get similar results.

Table 5: Detection rate under simple case

trace	Detection rate (CTD)	Detection rate (ITD)
port010	100%	100%
port021	100%	100%
port026	100%	100%

From Table 5, the proposed schemes achieve good performance in terms of detection rates under the simple case. This is not surprising, and the reason behind is that, usually TCP port 8 is not a common port, and connections of 22 IPs on port 8 are also rare. As nodes used for bots are independent of normal nodes in the trace, the behaviors generated by the Nugache bot should deviate from any normal behavior clusters greatly, resulting in sufficient evidences for detection. On the other hand, the results in this simple case only provide limited support for evaluation of the proposed schemes, as it is very easy to design a series of variation of Nugache bot in a way such that each bot can choose any port used by normal users in the same network for C&C. And it is not necessary that two bots share the same port for C&C (e.g. one can choose TCP port 80; another can

choose TCP port 21, etc.). Moreover, usually the nodes generating bot traffic should not be independent of nodes generating normal traffic (e.g. one node can generate both bot and normal traffic from bot program and normal user respectively). The Nugache bot of this kind is much smarter and stealthier than the original one; it is also more practical in implementation. Therefore, we believe experiments of detecting such bot should be considered for a thorough evaluation.

4.3.2 *Detection in Realistic Case*

Since there are no traces available for the more advanced Nugachebot discussed above, we simulate the Nugachebot C&C behavior and add it to the normal nodes to evaluate the performance of the proposed approaches. In detail, we consider the nodes generating normal traffic also generate a small amount of P2P botnet C&C traffic. The P2P botnet traffic at each node is generated to a port randomly chosen from the recorded ports in the training data, and the peers are also chosen at random from the destinations recorded in the training data. That is, we assume the bot knows the approximate range of destinations and services of normal traffic in its network, only communicates with peers and uses the port recorded in the training data for C&C. Mixing the bot traffic with normal traffic in each node means the bot traffic can hide itself in the normal traffic. In addition, we reduce the peerlist to be 10. By simulating the environments this way, we simulate the realistic case that is most favoring the P2P botnet.

In this realistic case, what we are interested is the alarm rate, which refers to detecting bot existence and generating an alarm. Because this is the most important step without which there is no chance to identify bots. Also in this case, it is much harder to identify the bot, as bot traffic is correlated with the normal traffic. Therefore, we evaluate the proposed schemes in this case using alarm rate and leave the identification of individual bots and detection rate as future work.

For each trace, we pick one time interval as new sample data, simulate P2P botnet behaviors on the nodes randomly selected from the network, and use the rest intervals as the training data. We run the simulation 1000 times for each interval, therefore, there are all together 12000 instances of the P2P botnet scenarios for a 2-hour long trace. We then calculate the alarm probability for each trace. We also set the P2P botnet behavior occurs in 90% 70%, 50%, 30% and 10% (randomly chosen) of the nodes to test the sensitivity of the proposed approach (each test runs 12000 times). Each bot will connect to 10 peers during C&C. Like the simple case, there are no frequent connections between any peers. Among all its peers, we assume there is 50% chance that a peer is online (a peer can be turned off, thus unavailable in the botnet). So on average, each bot contacts ten peers, and finds five online peers available for communication. Also, we use the most concise communications between peers, that is, each peer sends the command using TCP (or any protocol over TCP). So taking into account the setup phase, there are at least 3 packets sent for one command.

We consider trace port003 and port008 in this case as an example. Since the first behavior cluster is not stable in previous sections, we only consider the second and third clusters in this section. That is, we set $l = 2$ for ITD and $l = 2$ $r = 2$ for CTD. The alarm rates of CTD and ITD

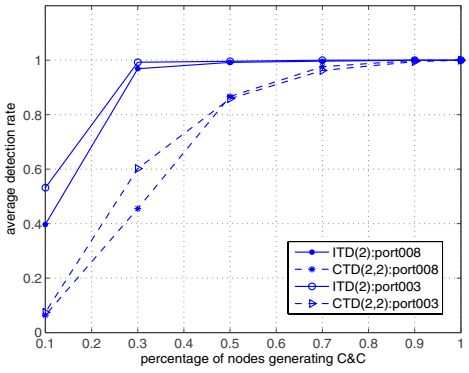

Figure 4: average alarm rate for port003, 008

in Fig 4 are very good ($> 95\%$) when the percentage of nodes being bots is over 70%. This indicates both schemes perform well when there are enough samples of bots available in the network. However, when the percentage of nodes being bots decreases from 70% to 30%, CTD based scheme decreases much faster than ITD based scheme, this indicates ITD is less affected by the percentage change at this interval. This also shows that ITD based scheme is much more robust than CTD, as CTD treats BPT and BMDT jointly, which is much more conservative. On the other hand, when the percentage is below 30%, both schemes decrease sharply, and is mainly because the bot samples are not enough to cause notable statistical changes on behavior clusters. A direct enhancement is to consider more popular behavior clusters during detection. On the other hand, from the perspective of the botmaster, since the botnet will be reused multiple times, to ensure the chance of being detected smaller than a certain value (e.g. less than 10%), the number of bots in the network where the proposed detection scheme (for example, CTD) is applied should be no more than 10% of the total nodes even if the botmaster has a good knowledge of the network where the bots are located.

4.4 Discussions and Future Work

As shown in the evaluation section, in the initial experiments, the proposed schemes achieve high performance in the simple case, and the performance in the realistic case of is also good (ITD is better than CTD). However, to achieve a more accurate detection in the realistic case, in addition to considering more popular behavior clusters, time effects (daytime or nighttime, weekday or weekend) should be further considered during behavior profiling and detection. As the behavior clusters will be quite different in different times, e.g. backup traffic is usually generated in the nighttime. On the other hand, a whitelist of normal but uncommon behavior profiles and correlating the ITD/CTD with the attack detection (e.g. A-Plane in [15]) will further help to reduce the false positive rates greatly.

Secondly, although the LBNL trace is the latest enterprise trace and widely used in many studies, it was captured during 2004 and 2005 and might be different from the current enterprise traffic. Therefore, we will consider more traffic traces for a thorough evaluation and design detection schemes for P2P botnet using P2P protocols if there is any new source of enterprise traces available. Last but not least,

for implementation of the proposed schemes, TCP traffic can be divided into P2P and non-P2P traffic, detection schemes of each traffic part can be further implemented. We will consider the above in our future work.

5. CONCLUSION

In this paper, based on the observation of correlations in node behaviors at different times, we designed algorithms based on statistical tests on popular behavior clusters to check if there is any unseen subtle activity from C&C in P2P botnets using non-P2P protocols. In the initial evaluations, we validated the assumption considered and achieved good performance in terms of high detection and low false positive rates and the results are encouraging for further research.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0627409, the Center for Information Protection, and by the Iowa State University Information Assurance Center.

6. REFERENCES

[1] http://en.wikipedia.org/wiki/internet_bot.

[2] http://www.honeynet.org/papers/bots.

[3] A. M. Antonopoulos. Security predictions for 2009, http://www.computerworld.com.au/.

[4] M. Bailey, E. Cooke, D.Watson, F. Jahanian, and N. Provos. Hybrid honeypot architecture for scalable network monitoring. Technical report, October 2004.

[5] C. A. J. Bambenek. Botnets: Proactive system defense. In *RO-DARPA-DHS Special Workshop on Botnets*, 2006.

[6] S. Chang and T. E. Daniels. Correlation based node behavior profiling for enterprise network security. In *SECURWARE*, 2009.

[7] S. Chang, L. Zhang, Y. Guan, and T. Daniels. A framework for p2p botnets. In *International Conference on Communications and Mobile Computing*, Jan, 2009.

[8] M. Collins, T. Shimeall, S. Faber, J. Janies, R.Weaver, M. D. Shon, and J. Kadane. Using uncleanliness to predict future botnet addresses. In *IMC*, 2007.

[9] E. Cooke, F. Jahanian, and D. McPherson. The zombie roundup: Understanding, detecting, and disrupting botnets. In *SRUTI*, 2005.

[10] D. Dagon, G. Gu, C. Lee, and W. Lee. A taxonomy of botnet structures. In *ACSAC*, 2007.

[11] D. Dagon, C. C. Zou, and W. Lee. Modeling botnet propagation using time zones. In *Annual Network and Distributed System Security Symposium*, 2006.

[12] J. L. Devore. *Probability and Statistics for Engineering and the Sciences*. Brooks/Cole Publishing Company, 6th edition, 2003.

[13] D. Dittrich and S. Dietrich. P2p as botnet command and control: a deeper insight. In *Malware*, 2008.

[14] J. Goebel and T. H. Rishi. Identify bot contaminated hosts by irc nickname evaluation. In *HotBots*, 2007.

[15] G. Gu, R. Perdisci, J. Zhang, and W. Lee. Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Security*, 2008.

[16] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee. Bothunter: Detecting malware infection through ids-driven dialog correlation. In *Proceedings of the 16th USENIX Security Symposium*, 2007.

[17] G. Gu, J. Zhang, and W. Lee. Botsniffer: Detecting botnet command and control channels in network traffic. In *NDSS*, 2008.

[18] G. JB, S. V, and N. C. Peer-to-peer botnets: Overview and case study. In *Proc. of the 1st Workshop on Hot Topics in Understanding Botnets*, 2007.

[19] A. Karasaridis, B. Rexroad, and D. Hoeflin. Widescale botnet detection and characterization. In *USENIX Hotbots*, 2007.

[20] M. H. Kutner, C. J. Nachtsheim, j. Neter, and L. W. *Applied Linear Statisti.al Models*. McGraw-Hill, 2005.

[21] P. Laborge. Bot attacks could hide in voip traffic. In *www.securityfocus.com/print/brief/119*, 2007.

[22] L. Liu, S. Chen, G. Yan, and Z. Zhang. Bottracer: Execution-based bot-like malware detection. In *ISC*, 2008.

[23] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney. A first look at modern enterprise traffic. In *ACM/USENIX Internet Measurement Conference*, 2005.

[24] M. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *Internet Measurement Conference*, 2006.

[25] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *SIGCOMM*, September 2006.

[26] A. Ramachandran, N. Feamster, and D. Dagon. Revealing botnet membership with dnsbl counter-intelligence. In *SRUTI*, 2006.

[27] M. K. Reiter and T.-F. Yen. Traffic aggregation for malware detection. In *the Fifth GI International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2008.

[28] R. Schoof and R. Koning. Detecting peer-to-peer botnets. http://staff.science.uva.nl/delaat/sne-2006-2007/p17/report.pdf.

[29] L. Stinson and J. Mitchell. Host-based, run-time win32 bot detection. In *RO-DARPA-DHS Special Workshop on Botnets*, 2006.

[30] A. Strehl, J. Ghosh, and R. J. Mooney. Impact of similarity measures on web-page clustering. In *AI for Web Search*, 2000.

[31] R. Vogt and J. Aycock. Attack of the 50 foot botnet. Technical report, 2006.

[32] K. Wang, G. Cretu, and S. Stolfo. Anomalous payload-based worm detection and signature generation. In *RAID*, 2005.

[33] P. Wang, S. Sparks, and C. C. Zou. An advanced hybrid peer-to-peer botnet. In *HotBots*, 2007.

[34] S. Wei, J. Mirkovic, and E. Kissel. Profiling and clustering internet hosts. In *Proceedings of the 2006 International Conference on Data Mining*, June 2006.

[35] C. Zou and R. Cunningham. Honeypot-aware advanced botnet construction and maintenance. In *DSN*, 2006.