Bayesian Bot Detection Based on DNS Traffic Similarity

Ricardo Villamarín-Salomón and José Carlos Brustoloni
Department of Computer Science
University of Pittsburgh
210 S. Bouquet St. #6135, Pittsburgh, PA, 15260, USA
(rvillsal, jcb)@cs.pitt.edu

ABSTRACT

Bots often are detected by their communication with a command and control (C&C) infrastructure. To evade detection, botmasters are increasingly obfuscating C&C communications, e.g., by using fastflux or peer-to-peer protocols. However, commands tend to elicit similar actions in bots of a same botnet. We propose and evaluate a Bayesian approach for detecting bots based on the similarity of their DNS traffic to that of known bots. Experimental results and sensitivity analysis suggest that the proposed method is effective and robust.

Categories and Subject Descriptors

C.2.3 [**Computer Communication Networks**]: Network Operations – Network monitoring; D.4.6 [**Operating Systems**]: Security and Protection – Invasive software (e.g. viruses, worms, Trojan horses).

General Terms

Security

Keywords

Bot, botnet, Bayesian method, DNS, Intrusion detection

1. INTRODUCTION

Bots are forms of malware that combine characteristics of worms, viruses, and Trojan horses. Their main distinguishing feature is that they continue to receive commands from a botmaster throughout their life cycle.

Many botnets have centralized command and control (C&C) servers. These servers may have fixed IP addresses or, more commonly, domain names that botmasters can map to IP addresses dynamically using DNS. In such botnets, bots can be detected by their communication with hosts whose IP address or domain name is that of a known C&C server.

To evade detection, botmasters are increasingly obfuscating C&C communication, e.g., using fastflux or peer-to-peer (P2P) protocols. Fastflux consists in modifying proactively at a fast rate the addresses associated with C&C servers' domain names (single flux). For added resilience, addresses of the name servers of the C&C zone may also be proactively modified at a fast rate (double flux). The domain names used may also vary. These frequent changes may frustrate detection based on known addresses or names. In P2P botnets, bots receive commands from peers, similarly making detection more difficult.

Regardless of obfuscation, commands tend to cause similar activities in bots belonging to a same botnet. This paper evaluates the hypothesis that bots in a same botnet have similar DNS traffic, through which they can be distinguished from other hosts.

We assume that at least one bot in a botnet is known. Using a Bayesian approach, we then find other hosts with similar DNS traffic. Experimental results suggest that the method is effective and robust.

The rest of this paper is organized as follows. Section 2 describes the Bayesian method in greater detail. Section 3 explains our methodology, and Section 4 presents our experimental results. These results are discussed in Section 5 and compared to related work in Section 6. Finally, Section 7 concludes.

2. BAYESIAN METHOD

Let Q be the set of DNS queries made during a specific time period, D be the domain names queried during that period, and H be a set of hosts that have made at least one query $q \in Q$. Every host $h \in H$ has an associated set Q_h of DNS queries. The objective of the Bayesian approach is to calculate, given Q_h, the probability that host h is infected by a bot:

$$Pr(\textit{host } h \textit{ is infected} \mid Q_h)$$

Let B (blacklist) be a set containing domain names of known C&C servers. We include in a set H_{bl} any host h that queried a blacklisted name during a specified period:

$$H_{bl} = \{\, h \in H : h \textit{ queried } b \in B \,\}$$

Let D_I and D_N be respectively the sets of all domain names queried by hosts in H_{bl} and in $H\text{-}H_{bl}$. Note that D_I may contain names that are not blacklisted. Furthermore, let $D_{BL}=D_I \cap B$ and $D_U=(D_I \setminus D_{BL})$. These sets are illustrated in Figure 1.

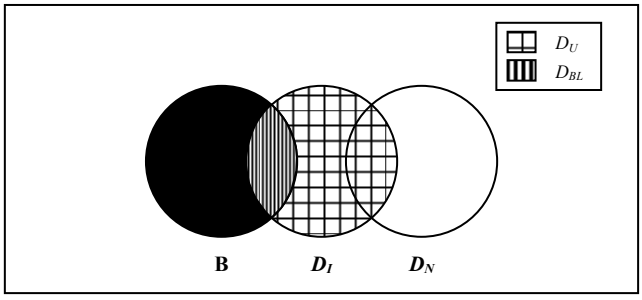

Figure 1. DNS names sets according to the hosts that queried them during the monitoring period.

Let Q_U be the set of queries for domains in D_U. A name $d_k \in D_U$ could be (1) a legitimate name that h's user queried during the monitoring period, (2) a legitimate name that the bot infecting a host $h \in H_{bl}$ queried to confuse detection tools, or (3) the name of an unknown C&C server.

SAC'09, March 8-12, 2009, Honolulu, Hawaii, U.S.A.

The problem is then to partition the set H-H_{bl} into two, namely, H_u, the set of uninfected hosts, and H_{sq}, the set of hosts that are infected but not in H_{bl} (e.g., because of C&C obfuscation by fastflux or other method). We achieve this partitioning in two steps.

First, we assign a score to every $q \in Q$ indicating a probability that a host making it is infected. The higher the score, the most likely is that the query is made by bot-infected hosts. We call *suspicious* the highest scored queries for names not in B. The set of such queries is Q_S. Likewise, we define the set Q_L of queries with the lowest scores.

Second, we assign to each host a score that combines the scores of all the queries it made. Finally, we assign each host to sets H_{sq} and H_u depending on the score received. If a host $h_1 \in H_{sq}$ is in the same botnet as host $h_2 \in H_{bl}$, h_1 and h_2 receive the same commands and therefore issue overlapping DNS queries. Then the set $Q_{h_1} \cap Q_S$ is expected to be larger than $Q_{h_1} \cap Q_L \cap Q_U$. In the remainder of this section we give details about how to perform this partitioning, including the criteria to consider a query or host score as high.

Let Ih_i denote whether a host h_i is infected and $|S|$ give the number of elements of a set S. Then:

$$P(q_j \mid Ih_i = 1) = \frac{|h_i \in I : q_j \in Q_{h_i}|}{|H_{bl}|}, \quad (1)$$

represents the likelihood that a query q_j is made by a host in H_{bl} [1]. However, even legitimate domains (e.g., www.microsoft.com) can be queried by these hosts and could receive a high value for $P(q_j \mid Ih_i=1)$. Therefore, the latter ratio is not appropriate to judge whether a query is suspicious or not. Thus, we also calculate the likelihood that a query q_j is queried by a host in H-H_{bl}:

$$P(q_j \mid Ih_i = 0) = \frac{|h_i \in (H - H_{bl}) : q_j \in Q_{h_i}|}{|H - H_{bl}|} \quad (2)$$

Then we obtain the posterior probability that a host h_i will send query q_j:

$$P(Ih_i = 1 \mid q_j) = \frac{P(q_j \mid Ih_i = 1) \cdot P(Ih_i = 1)}{P(q_j)} =$$

$$\frac{P(q_j \mid Ih_i = 1) \cdot P(Ih_i = 1)}{P(q_j \mid Ih_i = 1) \cdot P(Ih_i = 1) + P(q_j \mid Ih_i = 0) \cdot P(Ih_i = 0)}$$

It has been shown [13], and we have empirically verified, that assuming $P(Ih_i=1)=0.5$ gives good results under widely varying conditions. Making this assumption, then:

$$S_h(q_j) = \frac{P(q_j \mid Ih_i = 1)}{P(q_j \mid Ih_i = 1) + P(q_j \mid Ih_i = 0)}, \quad (3)$$

where $S_h(q_j)$ represents $P(Ih_i=1|q_j)$. Now, S_h can be calculated using equations 1 and 2, which have known values.

Equation 3 will produce an extreme value if a domain name is rarely queried. For instance, if the only host h querying the said domain belongs to H_{bl}, $S_h(q_j)$ will be 1 (and 0 if $h \notin H_{bl}$). We cannot consider as infected a random host querying this name in the future based only on this single evidence. This is because the query can be for a legitimate name even if it is sent by a host in H_{bl}.

To deal with these cases, we can use Bayesian statistics to determine $P(Ih_i=1|q_j)$. The calculation is based on observed DNS traffic as well as x, the a priori belief that a domain name that was never queried before will be queried by an infected host.

For this, we start assuming that Ih is a binomial random variable with a beta distribution prior, and that it represents the infection classification of a host making a specific query q [13]. Next, we perform n trials, each of which consists in examining the next host $h_i \in H$ that sent q to see if h_i is infected. If it is, we consider that the experiment was *successful* and assign $Ih_i=1$. This is a binomial experiment since (1) Ih can take only two values, 1 and 0, and (2) given two hosts, h_i and h_{i+1}, whether host h_i queried q is independent of whether host h_{i+1} also did so. For binomial experiments assuming a beta prior distribution, the probability that the n_{th}+1 trial will be successful can be expressed by [27]:

$$P(Ih_i = 1 \mid q) = \frac{\alpha + s}{\alpha + \beta + n}, \quad (4)$$

where α and β are the parameters of the Beta distribution, n is the number of trials and s is the number of successes involving q. However, we also want to incorporate in the calculation of the latter probability our a priori knowledge x. One way of doing this is to define $f=\alpha+\beta$ and $\alpha=f \cdot x$ [13], where f is a constant interpreted as the *strength* we want to give to x. Formula 4 then becomes:

$$S'_h(q_j) = \frac{f \cdot x + S_h(q_j) \cdot N_{q_j}}{f + N_{q_j}} \quad (5)$$

In this case s was approximated with $S_h(q) \cdot N_q$, where N_q is the total number of times a query q_j has been made during the traffic monitoring period. If we consider, for instance, values of 1 and 0.5 for f and x respectively, then a query q_j receives a neutral score $S'_h(q_j)=0.5$ before it is sent by any host. Thus we avoid the extreme values that S_h can take when we do not have enough data.

Once S'_h has been computed for all the queries made by a specific host, we calculate indicators of whether a host is infected. It has been shown [14][15] that robust indicators are obtained by taking the geometric mean of the host's most extreme $S'_h(q)$ values (closest to 0 and 1). The selection of these particularly high and low values is done using two thresholds, T_h and T_l:

$$m = \#\{q \in Q_h \mid S'_h(q) > T_h\} \text{ and}$$

$$I(h) = \begin{cases} 1, \text{ when } m = 0 \\ 1 - \sqrt[m]{\prod_{(q \in Q_h \mid S'_h(q) > T_h)} S'_h(q)} \end{cases} \quad (6)$$

$$k = \#\{q \in Q_h \mid S'_h(q) < T_l\} \text{ and}$$

$$N(h) = \begin{cases} 1, \text{ when } k = 0 \\ 1 - \sqrt[k]{\prod_{(q \in Q_h \mid S'_h(q) < T_l)} (1 - S'_h(q))} \end{cases} \quad (7)$$

$N(h)$ and $I(h)$ indicate how likely it is that a host is infected or non-infected, respectively. We define a combined score:

$$C(h) = \frac{N(h) - I(h)}{N(h) + I(h)} \quad (8)$$

The combined score will be between -1 and 1. Values close to 1 indicate that the host is infected while values close to -1 indicate the host is not. A value of 0 means that there is not enough evidence to support conclusively one classification or the other based on our data and background knowledge. For consistency

with the other scores, we modify $C(h)$ so that we can get a score between 0 and 1 [14]:

$$P(h) = \frac{1 + \left(\frac{N(h) - I(h)}{N(h) + I(h)}\right)}{2} \quad (9)$$

$P(h)$ indicates our degree of belief that a host is infected. Since it rarely gets extreme values close to 1 or 0, we need to select a threshold above which a host can be considered infected.

2.1. Application to Bot Detection

In this subsection we propose two ways to tune the Bayesian method for the task of bot detection.

First, given that blacklisted names can only be queried by infected hosts, S'_h can be made equal to 1 for queries in Q_{BL}.

Second, for any host $h \in H_{sq}$, it is desirable to include the score S'_h of its suspicious queries in the calculation of $I(h)$. One way to improve the chances of such inclusion is to pick appropriate values for parameters f and x. If we want the score $S'_h(q)$ of such queries to be at least equal to the threshold T_h for a particular value ω we can have:

$$S'_h(q) = \omega = \frac{f \cdot x + S_h(q) \cdot N_q}{f + N_q}$$

$$= \frac{f \cdot x + 1 \cdot 1}{f + 1}$$

$$\omega \cdot (f + 1) = f \cdot x + 1$$

$$f = \frac{1 - \omega}{\omega - x} \quad (10)$$

For example, if $T_h = \omega = 0.95$ and $x = 0.5$ (i.e., neutral), then $f = 0.11$.

3. METHODOLOGY

The application of the Bayesian method requires tuning the parameters described in the previous section (e.g., T_h and T_l). After tuning, we can employ the method on fresh data to verify the method's effectiveness.

In our experiments, we used two sets. The first set contains computers that we know with certainty to be infected. The other set contains hosts we confidently know to be uninfected. We collected DNS traffic of infected hosts. We ran variants of the same bot in computers under our control. If the bots running in this group of hosts queried known and unknown names of their C&C server, we detected the former if they were in a blacklist. However, we were interested in testing if the Bayesian method could detect the latter names. We did that as follows.

First, we picked m traces of our n infected hosts, and we altered their DNS packets by obfuscating any blacklisted names in them. We obfuscated names by appending to them a non-existent ccTLD (.nv) to each blacklisted name (in A and CNAME queries & answers). We refer to the infected hosts with altered traces as the *masked* hosts, and to the ones with unmodified traces as the *unmasked* hosts. Masked hosts cannot be identified as infected based on a blacklist. Second, we merged the traces of the masked, unmasked, and uninfected hosts. Finally, we applied the Bayesian method to the merged trace. We then observed (1) which uninfected hosts were classified as such, and (2) which masked hosts were identified as infected, based on non-blacklisted names that both masked and unmasked hosts queried. If the Bayesian method can successfully recognize that the masked hosts are infected, it can detect bots that use unknown C&C names, e.g., due to fastflux or other C&C obfuscation.

We describe in the following subsections the tasks needed for this analysis.

3.1. Blacklist and Bot Specimens

For one month, we acquired malware samples from MWCollect [11]. MWCollect is a distributed honeypot with sensors around the world. We also obtained daily from Shadowserver [18] a blacklist of known C&C servers. We considered for our experiments only those samples that were classified as a bot by at least one of the 32 antivirus programs maintained by VirusTotal [12]. From this universe, we selected executables that (1) had the same name, according to the Kaspersky antivirus, (2) contacted a same known C&C server, and (3) had distinct MD5 signatures. This process selected the two variants of two distinct bots that we used in our experiments. They are classified by Kaspersky antivirus as Backdoor.Win32.SdBot.cmz [6] and Net-Worm. Win32.Bobic.k [7]. Both SdBot and Bobic.k (also known as Bobax) are very popular botnet families [26][28][29].

3.2. DNS Data Collection

We collected DNS traffic from uninfected and infected hosts as described next.

We monitored DNS traffic generated by 89 PCs in instructional laboratories of our university during two 24-hour periods starting February 13, 2008. These PCs run Microsoft Windows XP and are used throughout the day by university students of different academic programs. Their configuration is tightly maintained with the latest antivirus definitions and OS and application updates. We did not find any blacklisted domain name or IP address in their traces. We labeled such traces CSL-1 (February 13-14) and CSL-2 (February 14-15). These are considered the *uninfected hosts*.

To collect DNS traffic from infected hosts, we used a sandnet based on Truman [2]. We modified Truman to allow malware to run unattended and on multiple platforms simultaneously. We also installed a DNS server (BIND [22]) on the sandnet gateway to allow bots under test to resolve domain names. We ran in this sandnet the bot specimens we selected for testing. The SdBot and Bobic.k variants were run during February 13-14 and February 14-15 respectively. We used 4 sandnet hosts during each 24-hour period. Table 1 below summarizes our setup.

Table 1. Sandnet setup for collection of bot traffic

Bot name	Variant	# of PCs	Dates (2008)
SdBot	V1	2	Feb. 13-14
	V2	2	
Bobic.k	V1	3	Feb. 14-15
	V2	1	

Hereafter we will refer to the traces that each bot executable generated as <bot name>-<variant number>-<host number>. For example, SdBot-V1-3 is the trace generated by SdBot's variant #1 that ran on the sandnet's third host. We captured four traces of each bot running concurrently on different computers. We found that the traces differ even between instances of the same bot version. Table 2 shows the number of unique domains queried in each trace.

Table 2. Number of DNS names queried per trace

Trace	Total DNS names queried
CSL-1	6636
CSL-2	4512
SdBot-V1-1	6
SdBot-V1-2	20
SdBot-V2-3	4
SdBot-V2-4	8
Bobic.k-V1-1	26
Bobic.k-V1-2	27
Bobic.k-V2-3	7
Bobic.k-V1-4	27

3.3. Test Traces

For each bot we created a set of four traces each containing one unaltered trace and three obfuscated ones. These traces were generated by the variants of the same bot in the different hosts where they ran (Table 1). We varied sequentially the trace left unaltered in each of those combined traces. Our intention was to see if the Bayesian method could identify the masked hosts as infected. We then merged each of those four combined traces with the traces of uninfected hosts that were monitored during the same time period as the respective bots (CSL-1 for SdBot and CSL-2 for Bobic.k). We will refer to those merged traces as *test traces*. For instance, the unchanged SdBot-V1-1 trace was merged with CSL-1 plus the modified versions of traces SdBot-V1-2 and SdBot-V2-{3,4}. We labeled *each* of these test traces simply by adding the suffix -T to the name of the bot trace that remained unaltered. In the latter example, illustrated in Figure 2, the test trace is labeled SdBot-V1-1-**T**. Its masked hosts are the ones numbered 2, 3 and 4. Collectively, we will refer to these sets of new traces as CSL-1-Sdbot-T and CSL-2-Bobic.k-T.

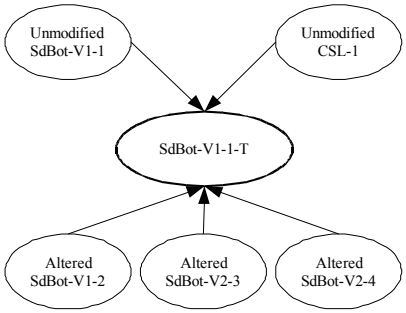

Figure 2. Example of creation of a test trace

Table 3 below shows the number of unique DNS names in each test trace, including the obfuscated ones. There were some common domain names in the traces of the bots and the uninfected hosts for both time periods. Also, the divergence in total names for traces in group CSL-2-Bobic.k-T is due to the varying number of obfuscated domain names in each trace.

Table 3. Number of DNS names queried in each test trace

Group	Test Trace	# Names
CSL-1-Sdbot-T	Any	6660
CSL-2-Bobic.k-T	Bobic.k-V1-{1,4}-T	4548
	Bobic.k-V1-2-T	4549
	Bobic.k-V2-3-T	4547

3.4. Evaluation metrics

To measure the effectiveness of our experimental classification we compute the resulting *recall* or True Positive Rate (TPR) [3], as defined by the formula:

$$TPR = \frac{TP}{TP + FN},$$

where *TP* is the number of true positives (infected hosts classified as infected) and *FN* is the number of false negatives (infected hosts classified as non-infected). The TPR has a maximum value of 1 if *all* the infected hosts are classified as infected, regardless of the number of misclassified uninfected hosts.

In addition, we calculate the resulting False Positive Rate (FPR) [3], as defined according to the following formula:

$$FPR = \frac{FP}{FP + TN},$$

where *FP* is the number of false positives (non-infected hosts identified as infected) and *TN* is the number of true negatives (non-infected hosts classified as non-infected).

4. EXPERIMENTAL RESULTS

Initially, we applied the Bayesian method only to the test traces in CSL-1-Sdbot-T (section 3.3). We wanted to find parameters that could yield good classification results with that trace, and then see if these same parameters were effective in trace CSL-2-Bobic.k-T. Following [1] we used values of $\omega=T_h=0.95$ and $P(I_h)=0.5$, and as in [20] we consider any host with a score $P(h)\geq 0.9$ (section 2) to be infected. Unlike [1] (and [15]) we do not use a value of T_l that is as far from the neutral score 0.5 as T_h is, because in our traces there are hundreds of queries with very low S'_h scores. This is illustrated in Figure 3 using as an example the test trace SdBot-V1-2-T. A value of $T_l=0.05$ would cause the inclusion of these low scores in the calculation of $N(h)$, thus overwhelming the contribution of the much less numerous queries with S'_h scores greater than T_h used to obtain the score $I(h)$. This would cause the resulting $P(h)$ scores to be much less than the classification threshold, negatively affecting the true positive rates.

We started with a very low value of $T_l=25\cdot10^{-6}$, which excluded from $N(h)$ all but the most popular query of any trace in CSL-1-Sdbot-T. Using these parameters we were able to classify all the masked hosts as infected without misclassifying any of the uninfected hosts. We later tried different values of T_l to evaluate the sensitivity of the method to such parameter. Starting with our initial T_l value, we tested new ones in increments of $\pm25\cdot10^{-6}$ until it was clear that trying further values was diminishing the accuracy of the classification. In total, we made 20 tests (4 test traces times 5 T_l values) for the first 24-hour period. We computed *TPR* and *FPR* values for every test and averaged the results of the traces in the same set for identical values of T_l (e.g., for $T_l=50\cdot10^{-6}$, all the rates for the traces SdBot-V<variant #>-<host #>-T were averaged.)

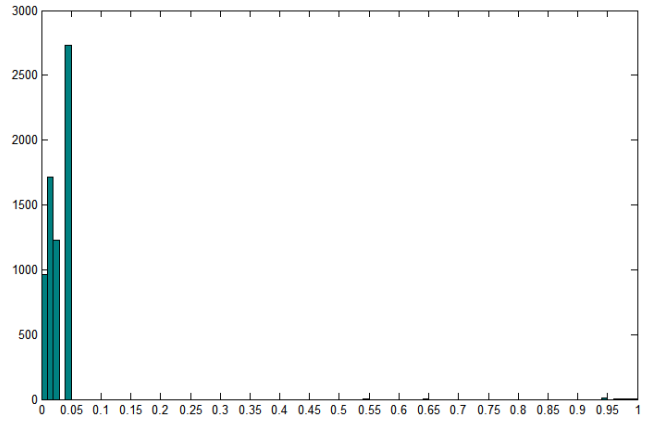

Figure 3. Histogram of S'_h scores in trace SdBot-V1-2-T

Table 4 summarizes the number of queries included in the calculation of score $N(h)$ for each value of T_l. The main reason the counts for T_l=100·10^{-6} differ in trace SdBot-V1-2-T from the others in CSL-1-Sdbot-T is that some of the domain names in these traces were not queried the same number of times by the bots, i.e., their N_q count was different. This affected the score $S'_h(q)$ for certain queries, since its calculation is sensitive to the value of N_q.

Table 4. Number of domain names with $S'_h < T_l$ considered in calculation of $N(h)$ for traces in CSL-1-Sdbot-T

Trace	T_l (millionths)				
	0	25	50	75	100
SdBot-V1-1-T	0	1	3	4	7
SdBot-V1-2-T	0	1	3	3	6
SdBot-V2-{3,4}-T	0	1	3	4	7

Figures 4 and 5 show averaged *FPR* and *TPR* values respectively. A confidence level of 95% was used to compute a confidence interval for the error bars. There were no false positives and no false negatives for T_l=25·10^{-6} and T_l=50·10^{-6}. The only instance in which we obtained one false positive was for trace SdBot-V1-4-T with T_l=0, when a misclassified host queried a domain that was also queried by the unmasked host in that trace. Such domain name belongs to an advertising firm [8][4] and was probably issued by the bot trying to confuse detection tools [1] (section 3). Further investigation by our university's tech support personnel found the host not to be compromised confirming it was indeed a false positive. Note that in this case the value of T_l was always less than any query's S'_h score and thus $N(h)$ scores for all hosts were equal to 1. This means the misclassified host was detected as infected based uniquely in the queries with top S'_h scores (the suspicious queries).

The cases when we obtained one false negative (a masked host identified as non-infected) are shown in Table 5. After inspection of these instances, we found that the culprit was the name *ad.doubleclick.net* which was queried by 0.87% (77 of 89) of the uninfected hosts *and* the only misclassified masked host. Hence, that query received a S'_h score less than T_l. This drove the host's $P(h)$ score considerably below the threshold (0.9).

After applying the method to CSL-1-Sdbot-T, we applied it to traces in CSL-2-Bobic.k-T. The parameters that gave no false positives for the first botnet also gave no false positives and no false negatives for the second botnet. To verify robustness of these results, we varied T_l. There were no false positives for any value of T_l tested. Figure 6 shows the averaged *TPRs* obtained. There were no false positives for any of the T_l values. However, in one instance we obtained no true positives either (i.e., the algorithm was unable to identify any masked host as infected.) This happened when T_l=100·10^{-6} in trace Bobic.k-V2-3-T, where the only unmasked host ran the second Bobic.k variant (V2) while the three masked hosts ran the first (V1). The cause of these misclassifications is explained next. Unlike Bobic.k-V2, Bobic.k-V1 queried the domain *www.microsoft.com* in all three PCs where it ran. During the corresponding time period, that was a popular domain among uninfected hosts (60 out of 89 queried it) and therefore it received an S'_h score below T_l. This in turn caused the $P(h)$ scores of the masked hosts to fall below our classification threshold (0.9). Table 6 shows the number of queries included in the calculation of score $N(h)$ for each value of T_l.

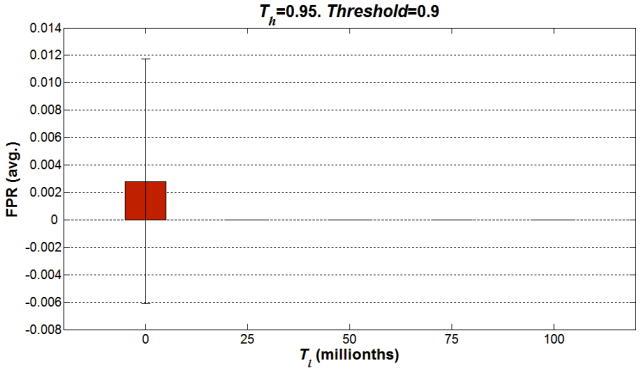

Figure 4. Average False Positive Rate for traces in CSL-1-Sdbot-T

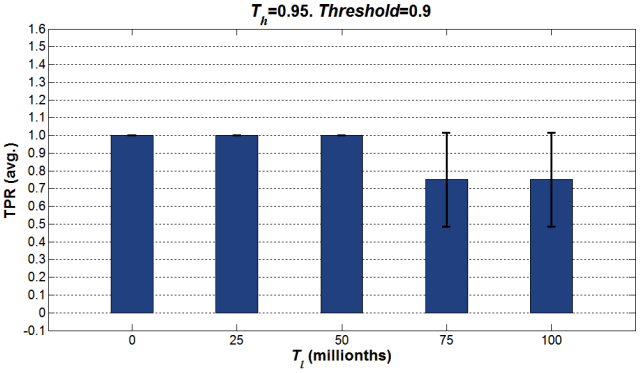

Figure 5. Average True Positive Rate for traces in CSL-1-Sdbot-T

Table 5. Cases when TPR < 1 for traces in CSL-1-Sdbot-T

Trace	T_l (millionths)	TP
SdBot-V1-1-T	75	2
	100	2
SdBot-V2-3-T	75	2
	100	2
SdBot-V2-4-T	75	2
	100	2

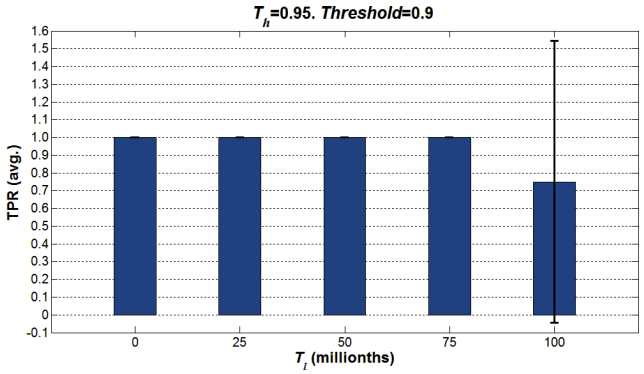

Figure 6. Average True Positive Rate for traces in CSL-2-Bobic.k-T

Table 6. Number of domain names with $S'_h < T_l$ considered in calculation of $N(h)$ for traces in CSL-2-Bobic.k-T

Trace	T_l (millionths)				
	0	**25**	**50**	**75**	**100**
Bobic.k-V1-{1,4}-T	0	1	2	3	3
Bobic.k-V1-2-T	0	1	2	3	3
Bobic.k-V2-3-T	0	1	2	3	4

5. DISCUSSION AND LIMITATIONS

Our results suggest the absence of false positives and false negatives for fairly wide ranges of parameters. However, if parameters are not well tuned, the method may generate false positives if a domain name is queried only by an infected host and one or a few of the uninfected hosts. In this case the method will incorrectly classify an uninfected host as infected because it queried a domain name that was not also queried by a sufficient number of other uninfected hosts. When we varied parameters for sensitivity analysis, the Bayesian method generated only one false positive. As explained before, the domain name that caused such misclassification did not correspond to a C&C server, but to a legitimate company (Microsoft). It was queried only by the bot and by the uninfected host deemed compromised by the method. Moreover, this specific instance occurred when no query's S'_h score was below the lower bound T_l. This error does not occur if T_l is properly tuned.

False negatives may occur, if parameters are not well tuned, when very popular domain names during a time period are queried by both infected and uninfected hosts. This may occur because the number of queries with a S'_h score near zero (indicating a low probability of the querying host to be infected) is usually considerably more than that of queries with scores near one. In our experiments, we observed false negatives only after we increased the lower bound threshold T_l to include more queries in the calculation of the host $P(h)$ score. Proper T_l tuning avoids this error.

6. RELATED WORK

The method used in this paper is based on a Bayesian technique to distinguish between spam and non-spam ("ham") e-mails, due to G. Robinson [13] (based on previous work by P. Graham [20]). It was employed to classify a new e-mail message as spam or ham based on the words it contains and the frequency those words appear in messages in two manually pre-classified sets. These sets consist of spam and non-spam e-mails respectively. Robinson modified the original method to allow the inclusion of any background information available about the probability of classifying a message as spam when words in it are seen for the very first time [13].

Ishibashi et al. [1] applied Robinson's method for the identification of a mass-mailing worm (Netsky [10]). They proposed to calculate the posterior probability of a host h being infected by the worm, based on the queries Q_h the host made during a specific traffic-monitoring period. A bug in Netsky caused it to make a distinctive, albeit bogus, query of an undefined DNS type. Such query was used as a signature for classifying a host as initially-infected. However in [1] they lacked a list of infected hosts (apart from those making the bogus query) in the traffic monitored, so the true detection rate achieved was not reported.

Rajab et al. [25] present a comprehensive infrastructure for botnet capture and tracking. It allowed them to perform an extensive investigation of measurements that characterize the structure and behavior of botnets. As part of their study they describe a life-cycle model of a typical botnet infection in terms of the bot activities such as download of the bot binary, C&C server DNS name resolution and C&C communication.

A similar model was presented by Gu et al. [24]. However they (1) exclude C&C name resolution and the monitoring of bot propagation in internal networks, and (2) include the bot activities of inbound scanning of internal vulnerable hosts as well as outbound infection scanning of potential victims. Their system, BotHunter, must be located on a network's egress position to monitor communication flows caused by bot actions (characterized in the life-cycle model). Bothunter uses anomaly detection techniques and a signature engine to identify those flows and then employs a correlation mechanism to issue an alarm if enough of the bot activities are detected. Unlike our approach, Bothunter (1) needs to monitor all network traffic, (2) does not use DNS traffic, and (3) performs more time consuming and complex activities [24].

BotSniffer [26] tries to discover spatial-temporal correlations and similarities in two types of bot responses to directives sent through a C&C channel. They are activity response (e.g., port scanning and spamming) and message response (e.g., informing the C&C of some action's outcome). In both cases, bots reply in a similar way and at a similar time window, forming a "response crowd" that exhibit strong synchronicity. BotSniffer uses anomaly detection techniques to identify bot infection by monitoring the two types of responses within a group (a set of hosts connecting to the same server). These techniques detect botnets respectively by analyzing (1) "dense crowds", i.e., those consisting of sizeable fractions of hosts in a group that issue message or activity responses, and (2) homogeneous crowds, i.e., those whose members have similar message responses. In contrast to these group analysis techniques, we only need to process DNS traffic.

Choi et al. [23] proposed two methods to identify botnets based on DNS traffic. The first computes a similarity between IP lists of hosts that made the same DNS query during two consecutive time periods. If the similarity is greater than a threshold, their system raises an alarm. For this they needed IP lists of at least five members. The second method detects multiple names of a C&C server if the IP lists of such names are of similar sizes. Nonetheless, the first technique can be defeated by spoofing IP addresses unless they maintain certain state information. This may

affect the method's scalability. The second technique can generate false positives if the IP lists of two legitimate domain names happen to be of comparable sizes without being excessively large. Our approach however, can identify groups of less than five infected hosts.

In another paper, we evaluated three heuristics for detecting anomalies in DNS traffic [30]. We found one heuristic to be particularly reliable: Names outside the querier's domain for which many DNS replies with NXDOMAIN status are observed usually correspond to C&C servers that have been taken down.

7. CONCLUSION

We proposed and evaluated a Bayesian method for botnet detection. Although similar methods have been broadly adopted by the spam-detection community and used before for worm detection [1], it had not been applied to bot detection before. Also, in the latter case no conclusive information about its effectiveness was reported. In this study, we found that the technique successfully recognized C&C servers with multiple domain names, while at the same time generating few or no false positives. Our sensitivity analysis suggests that the method is robust.

8. ACKNOWLEDGEMENTS

This project was funded in part by The Technology Collaborative through a grant from the Commonwealth of Pennsylvania, Department of Community and Economic Development.

9. REFERENCES

[1] K. Ishibashi, T. Toyono, K. Toyama, M. Ishino, H. Ohshima, I. Mizukoshi, "Detecting MassMailing Worm Infected Hosts by Mining DNS Traffic Data," *ACM Symposium proceedings on Communications architectures and protocols (SIGCOMM '05)*, pp 159-164, August 2005.

[2] J. Stewart. "Truman - The Reusable Unknown Malware Analysis Net." [Online] http://www.secureworks.com/research/tools/truma.html

[3] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, "Introduction to Data Mining" (1st ed.)

[4] McAfee © SiteAdvisor, "Report for cpaclicks.com," [Online] http://www.siteadvisor.com/sites/cpaclicks.com

[5] The Honeynet Project, "Know your Enemy: Tracking Botnets – Bot-Commands", [Online] http://honeynet.org/papers/bots/botnet-commands.html

[6] Kaspersky Lab's VirusList.com, "Backdoor.SdBot.gen" http://viruslist.com/en/viruses/encyclopedia?virusid=24976

[7] Kaspersky Lab's VirusList.com, "Net-Worm.Win32.Bobic.k", [Online] http://viruslist.com/en/viruses/encyclopedia?virusid=90085

[8] Shawn Collins' Affiliate Marketing Blog, "Florida Attorney General Investigates Affiliate Marketers," [Online] http://blog.affiliatetip.com/archives/florida-attorney-general-investigates-affiliate-marketers/

[9] F. Weimer. "Passive DNS Replication," in *Proc. 17th Annual FIRST Conf.*, July 2005. [Online] http://www.first.org/conference/2005/papers/florian-weimer-paper-1.pdf

[10] Kaspersky Lab's VirusList.com, "Email-Worm.Win32.NetSky.ae," [Online] http://viruslist.com/en/viruses/encyclopedia?virusid=50431

[11] MWCollect. "Malware Dedicated Whitehats." [Online] http://www.mwcollect.org/

[12] VirusTotal. "Free Online Virus and Malware Scan." [Online] http://www.virustotal.com/

[13] Gary Robinson. "A statistical approach to the spam problem". In Linux Journal 107, March 2003, [Online] http://www.linuxjournal.com/article.php?sid=6467

[14] Gary Robinson, "Spam Detection", [Online] http://radio.weblogs.com/0101454/stories/2002/09/16/spamDetection.html

[15] Greg Louis, "Bogofilter Calculations: Comparing Geometric Mean with Fisher's Method for Combining Probabilities," [Online] http://www.bgl.nu/bogofilter/fisher.html

[16] N. Ianelli and A. Hackworth. Botnets as a Vehicle for Online Crime. CERT Coordination Center, 2005.

[17] Evan Cooke and Farnam Jahanian. The zombie roundup: Understanding, detecting, and disrupting botnets. In *Steps to Reducing Unwanted Traffic on the Internet Workshop*, 2005.

[18] Shadowserver Foundation. [Online] http://shadowserver.org/wiki/pmwiki.php?n=Shadowserver.Shadowserver

[19] Honeynet Project. "Know Your Enemy: Fast-Flux Service Networks." [Online] http://www.honeynet.org/papers/ff/fast-flux.pdf

[20] Paul Graham, "A Plan for Spam," [Online] http://www.paulgraham.com/spam.html.

[21] Jonathan Zdziarski,. "Ending Spam: Bayesian Content Filtering and the Art of Statistical Language Classification". No Starch Press, 2005.

[22] Paul Albitzand and Cricket Liu, "DNS and BIND". O'Reilly and Associates, 2001.

[23] Hyunsang Choi, Hanwoo Lee, Heejo Lee, Hyogon Kim, "Botnet Detection by Monitoring Group Activities in DNS Traffic," in *7th IEEE International Conference on Computer and Information Technology (CIT)*, 2007.

[24] G. Gu, P. Porras, V. Yegneswaran, M. Fong, W. Lee: "BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation. In *Proc. of USENIX Security Symposium*, Boston, MA, August 2007.

[25] M. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. "A multi-faceted approach to understanding the botnet phenomenon". In *Proceedings of ACM SIGCOMM/USENIX Internet Measurement Conference*, Brazil, October 2006.

[26] G. Gu, J. Zhang and W. Lee. "BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic," in *Proceedings of the 15th Annual Network and Distributed System Security Symposium, ISOC*, February 2008.

[27] David Heckerman. "A tutorial on learning with Bayesian networks." In Michael Jordan, editor, *Learning in Graphical Models*, pages 301–354. Kluwer Academic, 1998.

[28] M. K. Reiter and T.-F. Yen. "Traffic aggregation for malware detection." In *Proceedings of the Fifth GI International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'08)*, 2008.

[29] Inoue, D. Yoshioka, K. Eto, M. Hoshizawa, Y. Nakao, K. "Malware Behavior Analysis in Isolated Miniature Network for Revealing Malware's Network Activity". *IEEE International Conference on Communications (ICC)* 2008.

[30] Villamarín-Salomón, R., Brustoloni, J.C. "Identifying Botnets Using Anomaly Detection Techniques Applied to DNS Traffic". *5th IEEE Consumer Communications and Networking Conference (CCNC)*, 2008.