Wide-scale Botnet Detection and Characterization

Anestis Karasaridis, Brian Rexroad, David Hoeflin

Abstract— **Malicious botnets are networks of compromised computers that are controlled remotely to perform large-scale distributed denial-of-service (DDoS) attacks, send spam, trojan and phishing emails, distribute pirated media or conduct other usually illegitimate activities.**

This paper describes a methodology to detect, track and characterize botnets on a large Tier-1 ISP network. The approach presented here differs from previous attempts to detect botnets by employing scalable non-intrusive algorithms that analyze vast amounts of summary traffic data collected on selected network links. Our botnet analysis is performed mostly on transport layer data and thus does not depend on particular application layer information. Our algorithms produce alerts with information about controllers. Alerts are followed up with analysis of application layer data, that indicates less than 2% false positive rates.

I. Introduction

MALICIOUS botnets are networks of ”bots”, compromised hosts that are remotely controlled by a master host via one or more controller hosts. The master host is the computer used by the perpetrator and is used to issue commands that are relayed to the bots via the controllers. The controllers are often Internet Relay Chat (IRC) servers [1], which are normally used for relaying messages among client terminals. Controllers are often created from compromised hosts that perform a coordinating role for the botnet. Figure 1 illustrates at a high level the relationship between the master, a controller, the bots and potential targets.

Botnets are used for various purposes, most of them related to illegitimate activity [2,3]. Some of their uses include launching distributed denial-of-service (DDoS) attacks, sending spam, trojan and phishing email, illegally distributing pirated media, serving phishing sites, performing click fraud, and stealing personal information, among others. They are also the sources of massive exploit activity as they recruit new vulnerable systems to expand their reach. Botnets have developed several techniques in their malware and infrastructure that make them robust to typical mitigation techniques. Due to their sheer volume, diverse capabilities, and robustness they pose a significant and growing threat to the Internet as well as enterprise networks. The threats undermine the reliability and utility of the Internet for commerce and critical applications, and therefore, better understanding of the structure of individual botnets is needed to formulate appropriate mitigation strategies.

Previous analysis [4] has shown that the majority of botnets have been traditionally based on IRC. This is due to the ability of IRC to easily scale to thousands of clients. There are existing cases of other types of botnets based on HTTP, DNS, and peer-to-peer models. In [5], several heuristics are suggested to identify possible IRC controllers by looking at servers with abnormal ratios of invisible to visible users, counts of users per channel or unusual channel user and nick names, or service ports. Use of flow data to detect idling clients to typical IRC ports is suggested in [6] as a way to identify bots. In [7], it is suggested that the use of secondary bot behavior such as propagation and attacks should be used instead of trying to detect control traffic directly. DNS black lists are frequently used by spammers to find which of their hosts are black listed and analysis of such queries can lead to additional bot identification [8]. Other detection approaches involve the setup of honeypots and the analysis of the captured malware to identify controllers [3,9].

The contribution of our work is the development of an anomaly-based passive analysis algorithm that has been able to detect IRC botnet controllers achieving less than 2% false-positive rate. The algorithm is able to detect IRC botnet controllers running on any random port without the need for known signatures or captured binaries. Even though this analysis is tuned to IRC-based botnets, we believe botnets will continue to require inventory management as well as a command and control structure that allows the botnets to be detected using similar methods. There are some distinct advantages to this type of botnet detection: *a)* Network data analysis is entirely passive, so it is invisible to the botnets, it does not interfere with network operations, and it does not run any risk of contributing to the problem. *b)* With some investment, it has been shown to be scalable to very large networks, which actually provide larger data sets for correlation of activity. *c)* It is able to show the dynamics of botnet activity by detecting activities that have been most effective in targeting our specific customer sets. We believe this analysis complements honeypot-based analysis. This work is being performed as part of the product evolution for AT&T Internet Protect [10] and other aspects of AT&T's network security services portfolio.

The rest of the paper is organized as follows: Section II describes our data collection process. Out botnet controller detection algorithm is presented in Section III. The algorithm for botnet client classification is presented in Section IV. Some quantitative results on the number of controllers and bots we have detected is given in Section V. Finally, we present our conclusions and ongoing work in Section VI.

II. Data Collection and Format

In our analysis, we use primarily transport layer flow summary data for identification of botnet controllers. In comparison to packet-level analysis, flow data reduces some privacy protection concerns. Flow data also significant reduces the

A. Karasaridis and D. Hoeflin are with AT&T Labs, Middletown, NJ. C.B. Rexroad supports the AT&T Chief Security Officer, Florham Park, NJ.

amount of data to process, which makes it practical to transport data to a central location for cross-correlation. The implementation is scalable to large networks in comparison to packet-level analysis since nearly all network devices can generate at least sampled flow data without significant performance impact or modification. In our application, we have invested in the capability to generate unsampled data in select portions of the network to complement more sparsely sampled flow data across a large Tier 1 ISP network. This implementation collects many billions of flow records each day for security analysis in addition to botnet identification processing described here.

Flow records contain summary information about sessions between a single source address/port (`sip`/`sport`) and a destination address/port (`dip`/`dport`) using a given protocol. A single flow record contains the number of packets, bytes, and an OR function of the flags used (if TCP is the transport layer protocol), the start and end time of the session and the transport layer protocol used. Flow record data are collected from a large number of geographically and end-point diverse circuits to a central processing facility where they can be filtered, processed, and correlated.

In the following section we describe the botnet controller detection algorithm.

III. Detection of Botnet Controllers

At a high level, our approach to detect botnet controllers consists of the following steps below.

1) Aggregate triggers, identify hosts with suspicious behavior (i.e., suspected bots) and isolate flow records to/from those hosts.
2) Analyze flow activity to identify candidate control flows and summarize them in conversations.
3) Aggregate and analyze candidate control conversations to isolate suspected controllers and controller ports.
4) Post reports and alarms.

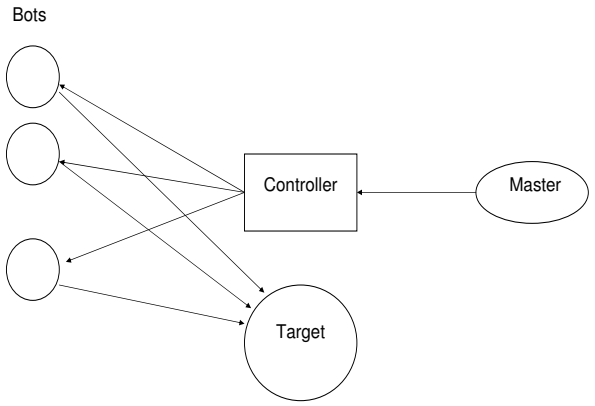

Fig. 1. Relationship between the bots, a controller, the master and the potential targets.

In Figure 2, we show a high level flow diagram of our algorithm for detection of botnet controllers. In the following paragraphs, we discuss in more detail the steps of the algorithm.

A. Aggregation of trigger events, identification of hosts with suspicious behavior, and selection of flows

Our analysis uses as input reports of suspicious host activities such as scanning, emailing of spam and viruses, or DDoS traffic generation. These reports are generated by internal upstream systems and identify the hosts that were involved in such activities, the type and duration of the activity, and the links where such activity was detected [11]. Scanning reports are processed further to identify interesting patterns, such as ports scanned by a large number of hosts, or ports for which we have seen an increase in the number of probes or the number of hosts probing them [12]. Records of spamming hosts are generated by analysis of SMTP logs stored in our inbound mail gateways and are subdivided into lists of hosts that send spam and hosts that send email viruses. Generation of such reports constitutes a trigger event that initiates and seeds our analysis. Our implementation of the algorithm allows additional types of trigger events to be incorporated.

The next step is to aggregate the trigger events and to search and fetch the flow records where the set of suspected bots appear as the source or destination of the traffic. In our implementation, we use empirical measurements to help determine which circuits are likely to show flow activity to or from particular IP addresses in specific time periods. This can help focus our search when many circuits of flow data are being processed.

Once the flow records of the suspected bots are fetched, we begin the process of selecting flow records that represent connections to potential controllers. These flow records are summarized to what we call *candidate controller conversations*. To identify these conversations we use three independent approaches which are described in the following paragraph.

B. Identification of candidate controller conversations

Due to a variety of factors, bots may start or end connections with their controllers at diverse times throughout the day. Consequently, we may not see a large number of flows generated simultaneously from a large proportion of bots. This is particularly true if the botnet is not receiving command activity. In these cases, long-term analysis help to identify controllers and/or generate higher levels of confidence in the detection. In order to do this, certain communications between suspected bots and other remote hosts are summarized and recorded individually. The information stored includes the remote port used and the number of flow records, packets and bytes transferred between the suspected bot and the remote host.

Here, we define ”conversation” as a summary of flow records between a local and a remote host on a particular remote port. A *candidate controller conversation* (*CCC*) is a conversation between a suspected bot and a remote host that satisfies certain criteria that are consistent with control

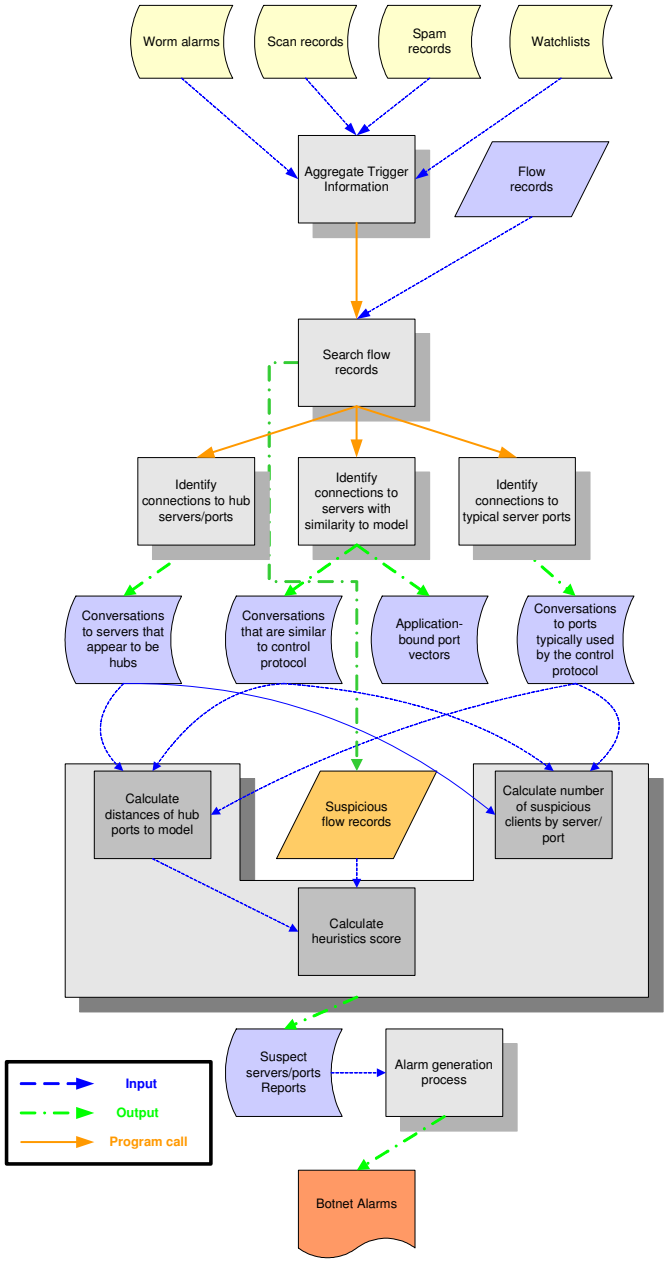

Fig. 2. Data flow diagram for the botnet analysis system.

traffic. Identification of *CCC*s involves initially the collection of all flow records generated by the suspected bots. The objective for isolating flows from the suspected bots is to allow correlation and modeling of suspected controller activity. In addition, the flow activity can also help to characterize activity associated with these suspected bots that may otherwise have been unnoticed. For example, additional ports used for attack vector may be identified. This information may be useful to help assess the threats posed by particular botnets.

Many controllers use ports typically associated with IRC, therefore these ports (e.g., 6667, 6668, 7000/tcp) are checked first to determine if there are obvious candidate controllers that

should be considered.

However, to help avoid detection, bots can use non-typical IRC ports to connect to the controllers. In many cases the control traffic is masqueraded as traffic from some other well-known application. For example Worm/Bot `W32.Spybot.ABDO` [13] uses port 53/TCP for its IRC communication with the controller. Note here that 53/TCP is typically used for large DNS transactions [14]. For this reason we use two other approaches to detect candidate controller flow records indicating connections between a suspected bot and a candidate controller on a non-typical port. The first such approach involves the identification of flow records between the suspected bots and remote hosts that appear to be hub servers, i.e., hosts which have multiple connections from many suspected bots to one or more of their local ports. Figure 3 illustrates this relationship where the hub server is on the left side of the figure and the suspected bots are on the right. In Figure 3, `lip` stands for local IP address, `lport` for local port, `rip` for remote IP address and `rport` for remote port. We are looking for pairs of `lip/lport` that have associations with multiple `rip`s or even with a single `rip` with multiple `rport`s. We build these associations by processing flow records of suspected bots in periodic intervals. We consider also the case where the hosts whose total traffic is analyzed (e.g., hosts that were detected scanning), are also the hub servers, therefore we do the processing both on the source and destination IP addresses of the flow records. The processing of flow records consists of the following stages.

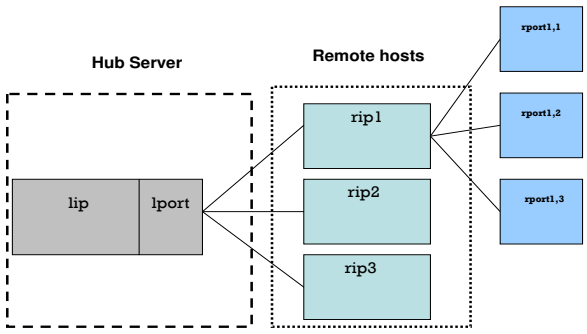

Fig. 3. Diagram of connectivity of a hub server.

- Capture all flow records related to a set of suspected bots.
- Store data in memory based on keys that involve the 4-tuples (`lport,lip,rip,rport`). The data include the number of flows, packets, bytes and last flow timestamp.
- Parse the data in memory and find pairs (`lport,lip`) that are associated with multiple `rip`s or `rport`s.
- Generate reports of such associations and store them on disk to correlate with later events.

A second approach to identify candidate controller connections to non-typical IRC ports is to find flow records between the suspected bots and remote servers which have traffic characteristics within the bounds of a flow model for IRC traffic. This model is constructed by periodically calculating percentiles of several metrics related to the flow records. These values are compared with reference models that represent typical command and control activity to determine likely CCCs. Each *CCC* record contains the following data for each suspected bot-candidate controller association:

- client IP address.
- remote server IP address.
- remote port.
- number of flows.
- number of packets.
- number of bytes.
- timestamps of first and last flows of the conversation.
- link where the activity took place.
- type of detection

C. Analysis of Candidate Controller Conversation records

The analysis of *CCC* records consists of three main parts: a) calculation of the number of unique suspected bots for a given remote server address/port, b) calculation of the distances between the traffic to remote server ports and the model traffic, and c) calculation of a heuristics score for server address/port pairs that remain candidates from (a) and (b).

After we calculate the number of unique suspected bots per remote server address/port in (a), we sort them by the number of suspected bots and apply a second derivative estimate to find the "knee of the curve". This is selected where the second derivative, estimated by second order differences, is maximized. If a server is above the "knee" point of the number of suspected bots curve, it is considered for further evaluation. This approach in general allows us to automatically focus on the larger botnets. Nevertheless, our implementation also gives the flexibility to consider server address/port pairs where the number of suspected bots is smaller than the "knee" point.

For the calculation of distance in (b) above, we use four statistics –0th, 25th, 50th, and 75th percentile– of the metrics flows-per-address (*fpa*), packets-per-flow (*ppf*) and bytes-per-packet (*bpp*) calculated from the flow records between all suspected bots and servers on a particular port. This approach allows averaging of the traffic characteristics across all suspected bots connecting to a remote port. The distance D_p of the traffic to server port p to the IRC traffic model is calculated as the average Euclidean distance of all statistics of the metrics between the observed and model traffic, i.e.,

$$D_p = \frac{1}{N_m} \sum_{i=1}^{N_m} \sqrt{\sum_{j=1}^{N_s} (X_{ij} - M_{ij})^2}, \qquad (1)$$

where $N_s = 4$ is the number of statistics, $N_m = 3$ is the number of metrics, and X_{ij} and M_{ij} are the observed and model traffic values of statistic j of metric i.

The port distances D_p are then sorted, and if a port distance is below a threshold, it is considered a candidate control port. Since this is early in the identification of controllers, the threshold is selected to minimize the false negative rate.

If a remote server address/port satisfies both conditions of large number of suspected bots and small traffic distance to the model, as described above, then it is analyzed in more detail where its flow records are processed to calculate a heuristics score.

One component of the heuristics score is the number of idle clients. Botnets tend to have a number of clients that are connected to the controller and listening for commands. These connections generate flow records that have certain patterns. For example, controlling IRC servers may generate *IRC Ping* messages to verify that their clients are connected. If the clients are connected, they are required to generate an *IRC Pong* message as soon as possible [1]. These messages generate flows that show periodic patterns for a given client-server pair.

Table I shows an example case where flow records show characteristics described above, namely TCP flow records between a client (source) and an IRC server (destination) having few packets and Push/Ack flags (flag value 24) where the flow arrival times show a period around 90 seconds. Since the client is the source in flow records, the flow records capture the *IRC Pong* messages. The bytes-per-packet ratio in this example is 65 Bytes but depending on the *IRC Pong* message this size can vary based on the parameters passed to the message (namely the server names that is addressed to).

In order to detect periodic patterns between suspected bots and hub servers, we use a hierarchical Bayesian model. We first separate flows for each client-server pair and sort them based on the Start time of the flow. We then calculate the interarrival times of flows as the difference between the beginning of the current flow and end of the previous flow.

```
model{
for (i in 1:N) {
  dt[i]~dnorm(mu[i],tau)
  log(mu[i])<-log(k2[i])+log(T)
  k2[i]<-round(k[i])
  k[i]~dunif(0.5,10.49)

}
T~dnorm(a,20)
a~dunif(85,480)
tau~dgamma(0.1,0.1)
s<-1/sqrt(tau) #stdev
cv<-s/T #coef var
}

#init:
list(T=88)

#data:
list(dt=c(94,90,93,90,180,91,93,91,89,91,
92,92,90,91,91,93,93,89,183,89,89,91,91,
91,91,89,89,92,88,94,182,98,91,88,91),N=35)
```

Fig. 4. Definition of a hierarchical Bayesian model used to detect and measure periodicity in flow interarrival times using the WinBugs language.

The interarrival data $dt[i]$ are modeled by normal distributions with means $mu[i]$ and precision[1] τ, where the means are multiples of a fundamental period T. Allowing the mean to be a multiple of a fundamental period enables the detection of the fundamental period even when multiple observations are missing. The number of observations that can be missing is controlled by the model $k[i]$. An example of a hierarchical Bayesian model used to detect noisy periodic sequences is described in Figure 4 using the WinBugs language [15]. In the example of Figure 4 we allow up to 10 missing observations. T is defined as a normal variable around a point a which is uniformly distributed in a region of typical periods. Initializing T to be the minimum or the 25-th percentile of the data contributes to the accurate convergence of the simulation.

Figure 5 gives the posterior densities of the estimated fundamental period T, the standard deviation s and the coefficient of variation C_v. The graphs illustrate that for the given data, the mean basic period is likely to be around 91sec and that the coefficient of variation is likely to be very small (0.02).

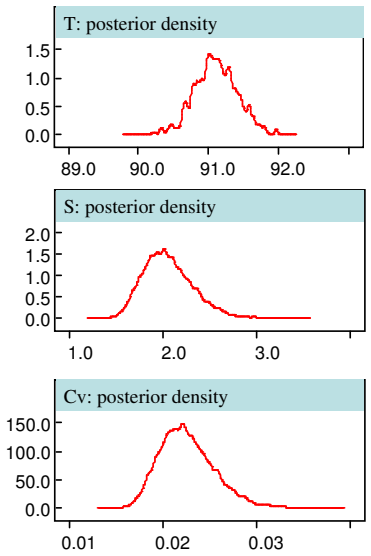

Fig. 5. Densities of the fundamental period T, the standard deviation s and the coefficient of variation C_v of T obtained by the hierarchical Bayesian model used to estimate the periodicity in the interarrival times of flows.

As an alternative way to detect quasi-periodic flow records that are incomplete (missing flows), we developed a modified K-means algorithm [16]. The algorithm follows the steps below.

1) Group flow records by (`sip`,`dip`,`sport`,`dport`) with at least an Ack flag for servers/ports that are deemed suspect based on port distance from the model and the number of connected suspected bots.
2) For each group of flow records calculate interarrival times of flow records (usually contain few packets).
3) Cluster interarrival times using the K-means algorithm.

[1]*Precision* τ is a function of the standard deviation s: $\tau = 1/s^2$.

TABLE I

FLOW RECORDS GENERATED BETWEEN AN IRC CLIENT (BOT) AND SERVER (BOT CONTROLLER) CAPTURING IRC PONG MESSAGES. A PERIODIC PATTERN IN THE INTERARRIVAL TIMES OF FLOWS IS EVIDENT.

Source IP	Destination IP	Packets	Bytes	Start Time	End Time	Source Port	Destination Port	Flags	Interarrival Time(sec)
a.b.c.d	q.w.e.r	2	130	1131397179	1131397179	55300	6667	24	
a.b.c.d	q.w.e.r	4	260	1131397273	1131397364	55300	6667	24	94
a.b.c.d	q.w.e.r	2	130	1131397454	1131397454	55300	6667	24	90
a.b.c.d	q.w.e.r	2	130	1131397544	1131397544	55300	6667	24	93
a.b.c.d	q.w.e.r	2	130	1131397634	1131397634	55300	6667	24	90
a.b.c.d	q.w.e.r	2	130	1131397725	1131397725	55300	6667	24	91
a.b.c.d	q.w.e.r	2	130	1131397818	1131397818	55300	6667	24	93
a.b.c.d	q.w.e.r	2	130	1131397909	1131397909	55300	6667	24	91
a.b.c.d	q.w.e.r	2	130	1131397998	1131397998	55300	6667	24	89

4) Examine if there are large clusters with a small coefficient of variation (CV). If yes, the algorithm terminates. The sample mean of the cluster is the approximate period.
5) Examine if there are clusters with means that are multiples of the basic period. If yes, this an indication of missing data and stronger evidence of the existence of a basic period.
6) Examine if there are clusters with small CV with means that are close. If yes, merge the clusters and recalculate the number of cluster members and the CV of the merged cluster. If the new cluster is large with a small CV, then use the new mean as the basic period and declare the client as periodic.

It is interesting to note here that this method can help identify control channels even when there are very few unique suspect bot-to-server communications available to evaluate.

The heuristics score is proportional to the fraction of quasi-periodic clients. Other considerations are made based on if the server uses both TCP and UDP on the suspect port and if the server appears to be serving significant peer-to-peer traffic (i.e., it has multiple peers on multiple service ports).

A confidence score is assigned to each suspected control server address and port based on the factors described above, i.e, number of suspected bots connected to the server, port distance, heuristics score, number of triggers, number of types of triggers, and alarm records are generated when the confidence score is above a threshold. The threshold can be adjusted based of the false positive rate that can be tolerated.

D. Validation of Controllers

Three methods are generally used to validate suspected botnet controllers:

- Correlation with other available data sources (e.g., honeypot based detection)
- Coordination with a customer for validation and mitigation.
- Validation of domain names associated with services (i.e., valid services generally have appropriately registered domain names with verifiable contact information).

IV. CHARACTERIZATION OF BOTNETS

One aspect of botnet characterization is the classification of the activities of its bots. Botnet operators often assign different sets of bots to perform different activities. For example, a set of bots may be responsible of recruiting other bots by scanning for certain vulnerabilities while another set is responsible for distributing email spam. However, while bots belonging to a certain botnet are expected to have some distinct modes of operation, individual bots are also expected to have unique behaviors due to variabilities in the software or hardware they run on, phase difference in their states, different background applications running simultaneously etc. One of our objectives in characterizing a botnet is to be able to classify the activities of its bots in the presence of background noise traffic. In this section, we describe a classification algorithm that creates and updates clusters of hosts based on their traffic profiles. This approach can be used for hosts belonging to a specific botnet or for any set of hosts that we want to classify based on its behavior.

Once we select the hosts we want to classify, we examine their traffic and calculate the number of flow records to application-bound ports. A destination port is considered application-bound if it is in the range of the IANA assigned ports (1-1023) [17], or there are at least two flows from the examined host to distinct remote addresses on a high-numbered port (1024-65535). A concise, yet descriptive, representation of the traffic profile of a host is a vector of application-bound ports ranked by the number of flows observed. These port-rank vectors are used as input to the classification scheme.

One important aspect of the classification scheme is the definition of a similarity function $S(i, j)$ between two vectors $\mathbf{v_i}$ and $\mathbf{v_j}$ representing signatures of the behavior of two hosts. Some of the desired properties of a similarity function are the following:

- $S(i, j) \in [0, 1]$.
- Similarity increases if a port number exists in both vectors.
- Similarity is a strictly decreasing function of the port rank.
- Similarity function is symmetric: $S(i, j) = S(j, i)$.

One function that satisfies all of the above properties is the following:

$$S(i,j) = \frac{\sum_{k=1}^{M} I_k(M - O_i + 1)(N - O_j + 1)}{N(N+1)(2N+1)/6}, \quad (2)$$

where M is the length of the shortest vector, N is the length of the longest vector, I_k is the indicator function that the port

with index k exists in both vectors $\mathbf{v_i}$ and $\mathbf{v_j}$. Finally, O_i and O_j are the orders in which the port with index k appears in vectors $\mathbf{v_i}$ and $\mathbf{v_j}$, respectively.

Example: Assume two vectors $\mathbf{x}_i = [445, 25, 53, 18067]^T$ and $\mathbf{x}_j = [25, 53, 135, 139, 445]^T$. Then the similarity value is calculated based on (2), where $M = 4$, $N = 5$ and $S(i,j) = (4-1+1)(5-5+1) + (4-2+1)(5-1+1) + (4-3+1)(5-2+1)/(5 \cdot 6 \cdot 11/6) = 0.491$.

The steps of our classification algorithm are the following.

- Given an initial set of hosts, calculate the similarity for each pair of hosts and rank the similarities with descending order. For the pairs with similarity larger than a threshold (e.g., 0.9), go to the next step.
- For each pair of hosts, check if any of them is already grouped (i.e., classified). If none of the hosts in the pair is grouped, start a new group. Calculate the traffic profile (vector of ranked application-bound ports) of the group. If one of the hosts is already grouped add the other host to the group.
- As new hosts are identified (in subsequent time intervals, e.g., 1 hour), calculate their similarity to all of the existing groups and allocate them to the group with the highest similarity above the threshold.
- If there is no group in which they can be allocated, allocate them to a common pool. Calculate similarities between all pairs of hosts in the pool and repeat the initial group formation process.

Below, we give an example of how the classification algorithm works.

Example: Assume that there no existing groups, and that we have identified hosts $A - E$ as targets for classification (e.g., suspected bots). We formulate their port-rank vectors and apply the similarity function for each pair. Assume that the similarity values are as shown in Table II. Since there are

TABLE II
EXAMPLE OF SIMILARITY VALUES BETWEEN A SET OF SUSPECTED BOTS $A - E$.

	A	B	C	D	E
A	–	1	0.91	0.1	0.2
B		–	0.8	0.3	0
C			–	0	0.7
D				–	0.97
E					–

no existing groups, the classification starts by examining the pairs with the highest similarity above the threshold, which in this example is equal to 0.9 (such threshold allows groupings of similar but not necessarily identical port-rank vectors): $S(A,B) = 1$, $S(D,E) = 0.97$, and $S(A,C) = 0.91$. Hosts A and B are currently not grouped and therefore we form the first group *Group1* with A and B. Then, we examine pair $D - E$. Since there is already one existing group, D and E are compared individually against the vector representing the traffic profile of *Group1*. Given that the similarity values of D and E to A and B are small or zero, D and E would form a new group, *Group2*. Pair $A - C$ has similarity 0.91, however A already belongs to a group, and C would be assigned to the same group as A, which is *Group1*. Assume now that in the next time interval three new suspected bots F, G, and H are identified and host C reappears. All hosts need to be compared to existing groups *Group1* and *Group2*. Assume now that F has high similarity to *Group1*, G and H have zero similarity to any of the existing groups and C now has strong similarity (above the threshold) to *Group2* (recall here that C was initially assigned to *Group1*). Then, F would be allocated to *Group1*, C's contribution to *Group1* would be removed and added to *Group2* while G and H would be added into the pool where we do pair-wise comparisons between unallocated suspected bots. If the similarity value between them is high and above the threshold, then a new group *Group3* would be created. Otherwise, they would not be considered for group allocation.

As mentioned above, when groups are formed, a group port-rank vector (equivalent to the group's traffic profile signature) is also calculated by aggregating the port ranks of the individual members that form the group. The new ranks are calculated based on the member ranks of the ports and the number of members that have contributed to a certain port rank. When a group is initially formed by two members the port is assigned a rank that is the average of the ranks of the port in the two members. When a host becomes member of a group, the ports are re-ranked based on the following criteria: *i)* The port exists in both vectors: The new rank of port k, R_k^{new} is calculated as follows.

$$R_k^{\text{new}} = R_k^{\text{old}} + \frac{r_k}{N_k}, \tag{3}$$

where N_k is the number of members contributing to the rank of port k in the existing group and r_k is the rank of the port of the new member to be added. *ii)* The port exists in the port-rank vector of the new member but not in the port-rank vector of the group. In this case the port is assigned a rank equal to a large number essentially putting it to the bottom of the rank of the group. *iii)* The port appears only in the group port-rank vector, in which case it maintains its rank after the merging of the new member. The final ranks are determined by sorting R_k^{new}. If the member is removed from a group the term $\frac{r_k}{N_k}$ in (3) is subtracted from the group rank term R_k^{old}.

Our classification algorithm allows the dynamic formation of groups (group expansion, shrinking, aging), tracking of group memberships and summarization of group profiles by group signatures. It is independent of the choice of the similarity function and robust in limitations in the data collection (e.g., one-way packet collection in flows, reduced number of collection points, etc.).

The algorithm is implemented in software that produces two files for each identified group and one master membership file for all the grouped hosts. The first group file contains summary information about the group such as the application-bound ports, the number of members of the group accessing a port and the time when the information was last updated. The second group file contains the IP addresses of the hosts belonging to the group, the ports that they accessed, and the last update time. The master membership file contains the IP addresses of the grouped hosts, the group that they belong to and the last update time. Examples of the output files are given

in Figure 6.

```
File: gma_10000001.txt

#Group Aggregate Port Information
#Port|Number_of_members|Last_updated
445|24|2005102614
25|24|2005102614
---
File: gap_10000001.txt

#Group Member Address Information
#IPaddress|Ports_ranked|Last_updated
a.9.184.67|445,25|2005100101
b.121.13.98|445,25|2005100101
c.140.212.26|445,25|2005100313
d.140.222.170|445,25|2005100412
e.166.113.174|445,25|2005100919
f.169.208.145|445,25|2005101002
.....
---
File: membdir.txt

#Member directory file
#IPaddress|Group|Last_updated
c.140.212.26|10000001|2005100313
b.121.13.98|10000001|2005100101
a.9.184.67|10000002|2005100101
g.132.126.130|10000002|2005102811
h.52.7.237|10000003|2005101521
i.52.2.205|10000003|2005102308
.....
---
```

Fig. 6. Sample output files generated by software that implements the host classification algorithm. *Note*: IP addresses are anonymized for protection of privacy. In this example, hosts belonging to *Group1* are all scanning for a known vulnerability on port 445 and have connections to port 25 to email spam or viruses.

V. Quantitative Results

Using our automated botnet detection system, we have detected 376 unique controller IP addresses between August 2006 and February 2007. Correlating these addresses with other sources (e.g., DNS data) has lead to the identification of several hundred additional controller addresses. Of these, 5 addresses were false positives since they were deemed non-malicious based on their DNS registration information. In the period between November 2005 to May 2006 we discovered 6 million unique IP addresses participating in malicious botnets and since then we have been discovering, on average, 1 million new bots per month. The botnets we have been seeing are very dynamic in nature: Based on long-term monitoring of validated malicious botnets, we estimate that the average bot stays about 2-3 days on the same botnet controller, switching controller addresses and domains very frequently.

VI. Conclusions and Future Work

We presented an algorithm for the detection and characterization of botnets using passive analysis based on flow data. Using our approach we have been able to detect several hundred controllers over a period of a few months running on arbitrary ports with very low false positive rate. Our approach has the following advantages: i) it is entirely passive and therefore invisible to operators, ii) scales to the largest of networks, iii) based on flow data analysis, which limits privacy issues, iv) has a false positive rate of less than 2%, v) helps identify botnets that are most affecting real users (and customers), vi) can detect botnets that use encrypted communications vii) helps quantify size of botnets, identify and characterize their activities without joining the botnet.

Our ongoing efforts focus currently in the following areas: a) integrate other forms of seed data into the analysis, b) expand the existing algorithms to accommodate better peer-to-peer and HTTP botnet controllers, c) automate DNS analysis to validate legitimate services, d) expand analysis to characterize structure and evolution of botnets.

Acknowledgment

The authors would like to thank Chaim Spielman, David Gross, and Daniel Hurley, of AT&T Security and Ken Futamura of AT&T Labs for providing valuable input.

References

[1] J. Oikarinen, D. Reed, "Internet Relay Chat (IRC) Protocol," IETF, Request for Comments (RFC) 1459, May 1993.
[2] N. Ianelli, A. Hackworth, "Botnets as a vehicle for online crime," CERT, Request for Comments (RFC) 1700, December 2005.
[3] The Honeynet Project & Research Alliance, "Know your enemy: Tracking botnets," http://www.honeynet.org, March 2005.
[4] K.J. Houle, G.M. Weaver, "Trends in denial of service attack technology," CERT, October 2001.
[5] J. Kristoff, "Botnets," NANOG32, October 2004.
[6] S. Racine, "Analysis of internet relay chat usage of ddos zombies," Master's thesis, ETH Zurich, April 2004.
[7] E. Cooke, F. Jahanian, D. McPherson, "The zombie roundup: Understanding, detecting and disrupting botnets," in *1st Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2005.
[8] A. Ramachandran, M. Feamster, D. Dagon, "Revealing botnet membership using dnsbl counter-intelligence," in *2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2006.
[9] M.A. Rajab, J. Zarfoss, F. Monrose, A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *Internet Measurements Conf. (IMC)*, 2006.
[10] AT&T, "Internet Protect," http://www.corp.att.com/internetprotect/.
[11] K. Futamura, "Method and apparatus for detecting scans in real-time," filed U.S. Pattent, December 2005.
[12] K. Futamura, W. Ehrlich, C.B. Rexroad, "Method and apparatus for detecting worms," filed U.S. Pattent, December 2005.
[13] Symantec, "Worm W32.Spybot.ABDO," http://securityresponse.symantec.com/avcenter/venc/data/w32.spybot.abdo.html, December 2005.
[14] P. Mockapetris, "Domain names: Implementation and specification," IETF, Request for Comments (RFC) 1035, November 1987.
[15] The BUGS project, "WinBugs," http://www.mrc-bsu.cam.ac.uk/bugs/welcome.shtml.
[16] D. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.
[17] J. Reynolds, J. Postel, "Assigned numbers," IETF, Request for Comments (RFC) 1700, 1994.