SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2015; **8**:2605–2623
Published online 28 February 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.500

RESEARCH ARTICLE

A graph-theoretic framework for isolating botnets in a network

Padmini Jaikumar* and Avinash C. Kak

Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906, U.S.A.

ABSTRACT

We present a new graph-based approach for the detection and isolation of botnets in a computer network. Our approach depends primarily on the temporal co-occurrences of malicious activities across the computers in a network and is independent of botnet architectures and the means used for their command and control. As practically all aspects of how a botnet manifests itself in a network—such as the online bot population, bot lifetimes, and the duration and the choice of malicious activities ordered by the bot master—can be expected to vary significantly with time, our approach includes mechanisms that allow the graph representing the infected computers to evolve with time. With regard to how such a graph varies with time, of particular importance are the edge weights that are derived from the temporal co-occurrences of malicious activities at the endpoints of the edges. A unique advantage of our graph-based representation of the infected computers is that it allows us to use graph-partitioning algorithms to separate out the different botnets when a network is infected with multiple botnets at the same time. We have validated our approach by applying it to the isolation of simulated botnets, with the simulations based on a new unified temporal botnet model that incorporates the current best understanding about how botnets behave, about the lifetimes of bots, and about the growth and decay of botnets. We also validate our algorithm on real network traces. Our results indicate that our framework can isolate botnets in a network under varying conditions with a high degree of accuracy. Copyright © 2012 John Wiley & Sons, Ltd.

KEYWORDS
botnet detection; botnet modeling; network intrusion; graph partitioning

***Correspondence**
Padmini Jaikumar, Robot Vision Laboratory, Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906, U.S.A.
E-mail: pjaikuma@purdue.edu

1. INTRODUCTION

Botnets are now considered to be a major security threat in computer networks. A botnet is a network of malware-infected machines controlled by a single entity called the "Botmaster." The malware itself may consist of viruses, worms, Trojans, and so on, basically any piece of software with networking and communication capabilities that was installed surreptitiously for the purpose of spamming, espionage, launching of distributed denial-of-service (DDoS) attacks, and so on [1]. A botnet may infect millions of computers. The Storm botnet is estimated to have infected 50 million. The botnet dismantled most recently, Rustock, was believed to have infected over a million computers; this botnet was sending billions of mostly fake-prescription-drugs related spam messages every day [2].

Detecting botnets in a network is the first step towards making a network free of such malware. Although it may be possible to bring to bear on the botnet elimination problem the more traditional virus scanning and removal tools even without knowing explicitly about the presence of a botnet, doing the same after a botnet has been properly detected and identified would allow for superior strategies to be used for getting rid of the malware and for the prevention of botnet re-infection. Additionally, detecting individual infected hosts may be harder than detecting the botnet as a whole because, in the absence of collective network information, the individual malicious activities must be detected with a higher degree of accuracy.

A key property that is exhibited by botnets is that of temporal correlations, in the form of co-occurrences, in their activities [3–6]. This property, which distinguishes botnets from other forms of malicious software such as isolated viruses and worms, is a key to designing an effective botnet detection system, as we show in this paper.

Although temporal co-occurrences in malicious activities at the infected computers—especially when such co-occurrences are analyzed in a proper statistical framework—facilitate botnet detection, the problem is, nevertheless, made challenging by the following reasons: as botnet architecture has evolved over the years, it has become more difficult to identify them by analyzing their communication patterns. Early botnets used the easier to recognize Internet Relay Chat (IRC)-based messaging for command and control (C&C). However, more recent botnets are based on Hypertext Transfer Protocol (HTTP)-based control (that is masked by the more routine HTTP traffic). Finally, when the bots use peer-to-peer (P2P) protocols—as is the case with the most advanced botnets today—for messaging and delivery of malware, that makes it harder still to identify the botnets through protocol-based traffic analysis [7].

Yet, another factor that makes detecting botnets difficult is that, of all the computers that are infected in a network, only a small and variable number may be online at any given time [6]. That different botnets in a network may be engaged in similar malicious activities also contributes to the difficulty of isolating the individual botnets [1]. The growth and decay of botnet malware have been compared to the spread of an epidemic [8,9]. At the beginning, the computers in a network may become infected at a high rate. However, as patches become available, the disinfection rates would also increase. Subsequently, the infection and the disinfection rates may compete to create large variations in the footprint of a botnet.

So, the challenge of designing an effective botnet detection tool boils down to, on the one hand, exploiting in a statistical framework the temporal activity co-occurrences that can be expected to exist between the different infected computers and on the other, avoiding the several difficulties mentioned earlier. Fulfilling that goal succinctly summarizes the contribution we have made in this paper. Our contribution here demonstrates how a graph-based framework for representing the infected nodes in a network—a framework in which the edge weights are calculated on the basis of the botnet activity co-occurrences at the endpoints—can be used to get around the problems that would otherwise arise when solutions are based on specific botnet architectures, specific activities, or on recognizing specific command-and-communication patterns.

Since the botnet activities at the infected computers can vary significantly with time, our graph-based representation is allowed to change with time. In particular, the edge weights that are derived from the temporal co-occurrences of the botnet activities at the edge endpoints change with time. On the one hand, this leads to a dynamic representation of a botnet in a network, and on the other hand, this allows the representation to be independent of the architecture of the botnet and of the protocols used for their command and control. Our botnet representation as derived from the pairwise temporal co-occurrences of potentially malicious activities is dynamic not only because it varies with time but also because it automatically allows for newly infected computers to be included in the representation and for the disinfected computers to leave the representation.

An important issue related to any new formalism for botnet detection is its validation—especially when a formalism is designed to capture the temporal behavior of a botnet (as ours is). A true statistically meaningful experimental validation would involve a controlled injection of botnets into a large network, placing bot monitors at each of the computers in the network and then demonstrating botnet detection and isolation with the proposed framework. However, a more practically feasible alternative is to simulate botnets according to the current best understanding of botnet behaviors and their growth and decay in networks. That is what we have carried out in our work. We have constructed a new unified botnet model that captures the current best understanding in the research community about how botnets behave, bot lifetimes, and the growth and decay of botnets. Our validation consists of demonstrating results on such simulated botnets. To add to this validation, we also show results on some real network traces.

To summarize, our contributions are as follows:

- We present a new graph-based framework for the isolation of botnets in a computer network.
- Our approach depends primarily on the temporal co-occurrences of malicious activities across the computers in a network, which makes it independent of the underlying botnet architecture.
- Our approach includes mechanisms to take into account the variations seen in botnet behavior over time, such as churn in the online bot population, bot lifetimes, malicious activity durations, and the growth and decay of botnets in the network.
- We validate the proposed approach by applying it to real network traces and to a simulated botnet model that captures the current best understanding of botnet behavior over time.

The rest of the paper is organized as follows: Section 2 discusses previous approaches to botnet detection. Section 3 discusses botnet behavior and the corresponding challenges in botnet detection in greater detail. Section 4 presents our graph-based framework for botnet detection. Section 5 presents the details of a new unified botnet model, which is used for validating our approach to botnet detection. Section 6 presents our botnet detection results on real and simulated network traces. Section 7 discusses some limitations of the proposed approach and possible ways to get around them. Finally, Section 8 concludes the paper.

2. RELATED WORK

There now exist several surveys that describe botnet architectures and botnet behaviors in considerable detail [6,8,10–13]. Whereas Stone-Gross *et al.* [10] and Holz *et al.* [11] have focused on the malicious activities that botnets typically engage in, Rajab *et al.* [6] have delved deeper into certain specific issues related to bot behavior, these being bot on–off times, bot migration, and so on. The reader's attention is also drawn to [9,14], and [15] where the authors have reported on the propagation patterns of malware in networks.

With regard to the literature that deals specifically with botnet detection, only a few techniques have been reported that are architecture independent [13]. A majority of the proposed techniques detect botnets that are centrally controlled with their communications based on the IRC protocol. As a case in point, Goebel and Holz [16] have proposed an approach for botnet detection that is based on identifying similarities in the IRC nicknames used by the hosts. For another example related to IRC-based botnets, Mazzariello [17] has shown how to detect the infected hosts by analyzing their IRC communications for the vocabulary of known botnets. For a different approach to the detection of IRC-based botnets, BotSniffer [3] focused on detecting the centralized C&C center of a botnet; subsequently, it tries to identify the bots as hosts that connect to the control center. As bots use encrypted communications and more importantly, with the evolving changes in botnet architecture, these and other IRC-based botnet detection approaches may no longer be useful.

As opposed to the work on IRC-based botnets, only a few approaches have been developed that try to detect the botnets that are based on the P2P architecture. Holz *et al*. in [11] described work that deals with disabling the P2P-based Storm worm. An approach to P2P botnet detection that is based on contact tracing was presented by Huang *et al*. [18]. BotGrep [19] is another technique where the authors attempted to detect P2P botnets by looking for specific C&C patterns that can be observed in the overlay topology.

There exists another category of botnet detection approaches that bases the detection decisions on the features extracted from network traffic logs and temporal correlations between these features across the different hosts in a network. These approaches tend to be free of assumptions regarding the architecture of the botnets and their communication protocols. To cite previous such contributions, the BotGraph framework [4] identifies spamming botnets as those hosts that show strong correlations across the hosts in their transmission of spam. Along the same lines, the BotMagnifier's framework [5] identifies a spamming botnet as a group of hosts that show temporally similar patterns of spamming behavior. In [20], the authors detected bots as hosts that exhibit temporal correlation in their Domain Name System (DNS) queries. BotHunter [21] takes a slightly different approach where the focus is on detecting individual infected bots as opposed to botnets—a bot is detected by recognizing the various stages that characterize a typical bot infection. A similar approach, also aimed at the detection of individual infected hosts, is adopted in [22] where behavior-based clustering is used to automatically analyze and classify Internet malware by using system state changes in terms of files written, processes created, and so on.

From the standpoint of being independent of the botnet architectures and making no assumptions about the communication protocols used, of particular interest to us is the BotMiner contribution [1]. In this approach, a botnet is detected by the following: (i) extracting features from the potentially malicious activity events at the hosts and their traffic logs; (ii) clustering the communication traffic and the activity events at the individual hosts in the feature space thus formed; and (iii) carrying out network-wide correlations of the clusters for the detection of similar malicious traffic and activities that characterize botnets. Clustering is also used in the work reported in [23] where the authors have demonstrated botnet detection by clustering the different types of application-layer traffic into the traffic generated by uninfected hosts and that generated by bots. Note that both these approaches do not take into account temporal co-occurrences of the potentially malicious activity events and traffic logs in the manner we describe in the rest of this paper.

To summarize, most of the existing techniques today are limited in what they can accomplish by way of botnet detection and isolation. Some of the existing techniques are limited by the fact that they focus on detecting only specific kinds of malicious activities that the hosts in a botnet may engage in, whereas some others are limited by their just trying to figure out whether a specific host in a network was infected by a botnet. The more recent work, such as that reported in [1], is more general, in the sense that it is neither limited to any specific botnet architecture nor based on any assumptions regarding their communication protocols and at the same time, it seems to identify the botnet as a whole in a network. However, it still does not fully exploit the information contained in the temporal variations in the malicious activities and the related traffic logs. We believe that what we need are approaches that, in a manner independent of the botnet architecture, not only include a wider repertoire of malicious activities in their analysis of the activity and network traffic data but also take into account temporal aspects of how the malicious activities, considered simultaneously, manifest themselves across the network. The work we present in this paper is a step in that direction.

3. BOTNET ACTIVITIES AND THEIR PROPERTIES

As mentioned previously, a *bot* is a self-propagating piece of code that infects hosts through various exploits—such as exploits based on buffer overflow and other similar vulnerabilities, exploits based on social engineering attacks that drop Trojans in the hosts, and so on [21]. In this regard, they are similar to viruses and worms. Bots, however, differ from isolated viruses and worms in the sense that they exhibit certain specific communication patterns and are externally controlled by a single entity that is commonly referred to as the *C&C server*. The communication patterns associated with bots reflect their ability to engage in coordinated malicious activities.

Detecting botnets is a key goal of network security administrators for reasons that are easy to understand. If a botnet can be identified and successfully isolated in a network, the same disinfection strategy can be quickly applied to all the hosts all at once. This considerably reduces the chance of reinfection from the still infected hosts when a network is cleaned up one host at a time. Network security admins are also generally of the opinion that if one could simultaneously detect all of the infected hosts participating in a botnet through an analysis of the temporal relationships between the potentially malicious activities exhibited by the hosts, that would be much more efficient than having to test each host in isolation for the presence or the absence of a botnet.

The goal of this paper, as described earlier, is to detect all the bots in each of the botnets that may have infected a network. The problem is challenging because of some unique features of bot behavior and growth patterns, which will be highlighted in the following discussion.

Figure 1 shows the functioning of a centrally controlled botnet, which is one of the most common botnet architectures. There are five bots in this botnet. At the instant that corresponds to the depiction in Figure 1, three of the five bots are *Online*, meaning that they are connected to the C&C center of the botnet. These three bots can be assumed to be actively receiving commands from the botmaster and responding to those commands. These three bots are identified in Figure 1 by using solid arrows with the label "Command." The other two bots in Figure 1 are *Offline* and therefore not currently available to do the botmaster's bidding. These two bots are identified in Figure 1 by using dashed lines with the label "Not Connected." At the instant to which the figure corresponds, the botmaster has commanded his or her bots to attack a computer that is shown as "Victim" in the figure. All the bots that are currently *Online* are engaged in the attack. The bots that transition from the state *Online* to *Offline* will cease performing the botmaster-dictated command, whereas the bots transitioning from the state *Offline* to *Online* will become involved in executing the dictated commands.

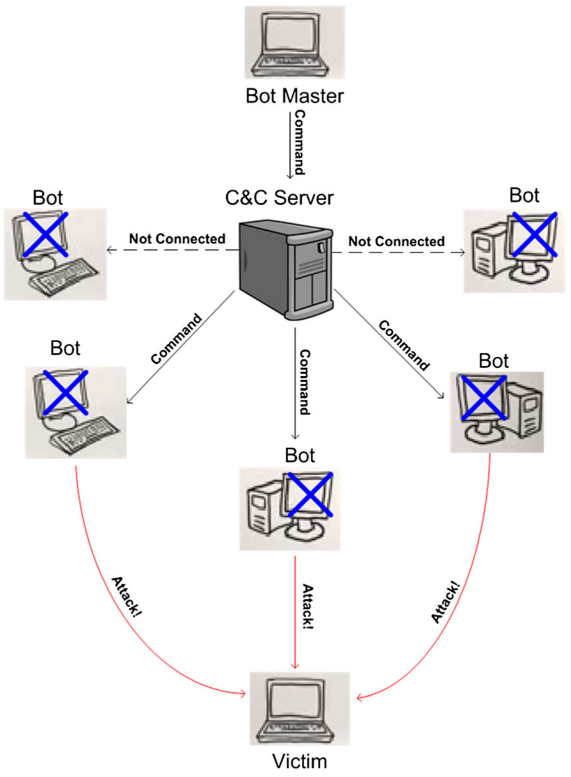


Figure 1. The functioning of a typical centrally controlled botnet.

3.1. Variable online footprint

It has been observed in networks that typically only a small and variable portion of a botnet is *Online* at any given time. This is because of a combination of factors: Bots may go offline because some folks want to turn off their computers at night; this may cause a botnet to exhibit a diurnal pattern in its functioning. Additionally and even more importantly, the bots in a network may be kept dormant for long periods to avoid detection and possible removal [6,24]. Figure 2 (from [6]) shows the estimated online footprint of a botnet whose total size was believed to be between 8000 and 10,000 hosts. As can be seen in the figure, only a fraction of the botnet was online at any given time. In addition, the online portion exhibited considerable variation in size. The fact that only a small fraction of a botnet may be active at any given time makes detecting the complete botnet extremely challenging.

3.2. Evolving architecture and communication patterns

A majority of botnets exhibit two distinct architectures—*centralized* and *P2P*. Centralized botnets, such as the one in Figure 1, usually employ the IRC protocol for C&C. The online bots of a centralized botnet receive the

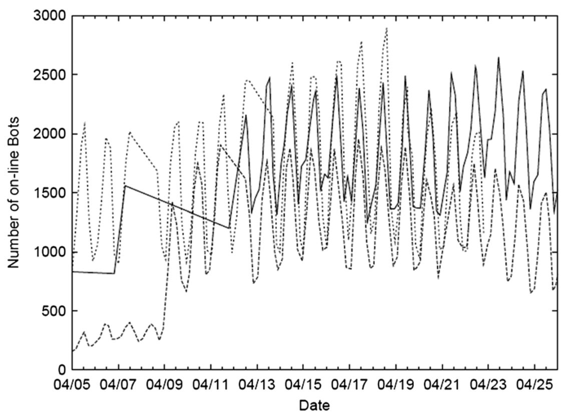


Figure 2. Variation in online footprint of a botnet (from [6]).

botmaster's commands instantaneously. This form of botnet communication has been referred to as the push mode for messaging the bots [1]. Alternatively, botnets based on the *P2P* architecture (such as the *Nugache* and the *Storm* worms) receive their commands in what may be referred to as the pull mode. In the pull mode, a botmaster publishes the command-bearing files in the P2P network of the botnet. Associated with the files are specific search keys known to the bots. Each online bot polls its neighbors/peers to locate the command files by using the search keys. Such a mechanism is used by the Storm botnet, which uses an unauthenticated publish/subscribe communication system [11].

In addition to the centralized and P2P botnets, we also have HTTP botnets. The bots in HTTP botnets query a designated web server for new commands every couple of seconds [3]. It is obviously the case that at any given time, only the bots of the botnet that are in the *Online* state can receive the botmaster's commands (in any architecture).

Given that there exist several different architectures for botnets (and the fact that these architectures are constantly evolving), it is imperative to not tie a botnet detection strategy to any specific architecture. Rather, the focus should be on exploiting the temporal co-occurrences in the activities exhibited by the bots of a botnet across the board (as an illustration of the usefulness of temporal co-occurrences, it is noteworthy that temporal correlations are used to identify the spam spewed out by P2P botnets in SNARE [25]). To give the reader an appreciation of what sort of bot behaviors we are talking about with regard to measuring temporal co-occurrences, we show in Table I typical botmaster commands along with their frequencies. The data in this table is based on the observations of over 192 IRC botnets (in [6]) and some P2P botnets (in [11]).

3.3. Multiple botnets in a network involved in similar activities

It is no longer uncommon for a computer network to be infected by multiple botnets. These botnets are generally engaged in similar activities, as observed in [6] and [3]. The relative frequencies of occurrence of the different activities can be assumed to be according to the data shown in Table I. Different botnets being engaged in similar activities adds to the challenge of isolating the botnets accurately. Figure 3, which is an example based on the observations of common malicious activities as reported in [6], presents the activities of three botnets in a network that are involved in similar activities.

Table I. Relative frequency of botmaster commands.

Command type	Frequency (%)
Control	33
Scan	28
Spam	15
Mining	7
Download	7
Attack	7
Other	3

The category "Other" stands for a grab bag of activities such as a non-server host acting surreptitiously as an HTTP server, or acting as an IRC relay server, and so on. Except for the row labeled "Spam," this table was taken from [6]. That row in [6] is labeled "Cloning." We believe that the choice of the label "Spam" reflects the current best understanding of the relative frequencies of botnet commands.

In the absence of directly taking into account the times of occurrence, the botnets are likely to be identified from network traffic logs on the basis of all of the entries up to the time a decision needs to be made about the presence or absence of botnet malware and the nature of that malware. Figure 4 shows the expected results at three different instants (marked as three *time slices* in Figure 3) regarding which specific botnets might have infected the network when only time-accumulated activity reports are taken into account. Just activity-based decisions using network logs from the start of the monitoring period up to *Time Slice 1* yield three distinct botnets because each botnet up to that time was involved in one distinctly different activity. On the other hand, the logs collected up to *Time Slice 2* (again from the start of the monitoring period) group the botnets 2 and 3 together because they were involved in a similar mix of activities up to that moment. Finally, by *Time Slice 3*, all the botnets look similar because the accumulated mix of activities for each is the same.

Therefore, any approach, such as the one used by BotMiner [1], that bases botnet identification decisions on just the accumulated activity reports in network traffic logs will yield different results at different points in time. It should be noted that, in the presentation of the results shown in Figures 3 and 4, we could not have stopped just after *Time Slice 1* for botnet identification decisions because, as noted earlier, the bots of a botnet come online on a discontinuous basis. One must therefore record the network traffic logs over a sufficiently long time to obtain a more realistic picture of the botnet activities in a network.

Figures 3 and 4 are presented to highlight the fact that one must also take into account the temporal variations in botnet activities, in addition to the activities themselves, in making botnet identification decisions.

3.4. Changing growth patterns

Botnets may grow or shrink over time. Because of many similarities between bots and viruses, it is reasonable to assume that the models that describe virus propagation [9,14,15] would also apply to botnets. Initially, when a new virus appears in a network, its growth is exponential. One would expect the same to be true for bot propagation, and then, as patches become available and as newer botnets appear, the footprint of the original botnet might shrink. Rapid variations in botnet infection sizes can create additional challenges for botnet detection.

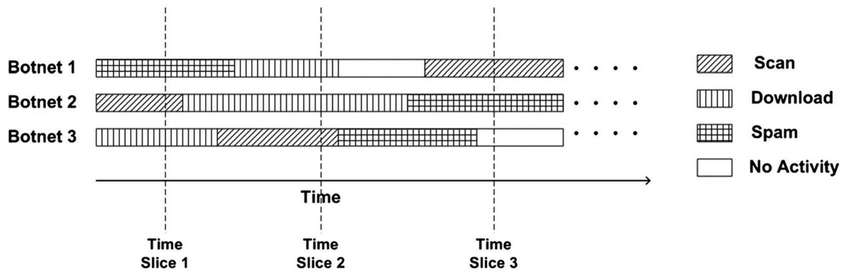


Figure 3. Overlapping botnet activities in a network.

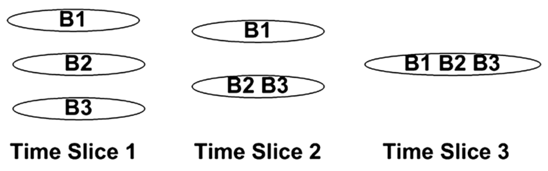


Figure 4. Identifying botnets just on the basis of accumulated activities as shown in Figure 3.

3.5. Changing botnet membership over time

The hosts in a network may go from being infected by one botnet to being cured and then to being reinfected by a different botnet. Therefore, the botnet label of a host may change with time.

It has been observed that at a given time, usually only one botnet is active in a host. Most botnets are resilient to takeover attempts by other botnets, and in the case that such a takeover happens, the new botnet makes sure to remove existing infections [11,26]. In [26], SRI researchers who analyzed the Conficker C worm observed that the worm prevented the other well-known botnet infections from attacking the host in which the worm resided.

The issues we have described in this section point to the challenges that must be addressed by a reliable botnet detection framework. On the strength of our understanding of these issues, in this paper, we propose a graph-based framework that has the potential to address several of these issues. We model the infected hosts in a network as a graph $G=(V, E)$ where V is the set of nodes that represent the hosts that have exhibited signs of malicious activity and E is the edges between the nodes, each edge representing the likelihood that the two nodes connected by the edge are part of the same botnet. Edges, updated periodically, encode both the activities and the temporal co-occurrences between them. Once the activities have been observed over a sufficiently long time (to allow for a large enough population of the bots to become active at some point) and we have updated the edges, we partition the graph by using the *Normalized Cuts* algorithm proposed in [27]. This partitioning assigns a bot label to each of the nodes in the graph, with each label corresponding to a different botnet. In this manner, we can identify the different botnets operating in a network and figure out which host is active in what botnet.

The combination of our edge update procedure and graph partitioning allows for the identification of the bots of the same botnet even when they are active at different instants. Our framework is also able to distinguish between the different botnets that may have been involved in the same activity at the same instant. Even more particularly, our approach does not require storage of network logs. At any point, all that needs to be stored is the updated graph. Hosts that become infected are added to the graph as new nodes, and the ones that stop exhibiting signs of malicious activity (and are reliably known to be free of infection) are removed from the graph.

The following section discusses the proposed approach in greater detail.

4. A GRAPH-BASED FRAMEWORK FOR BOTNET DETECTION

The goal of assigning an accurate botnet label to each infected host in a network is achieved by representing the infected hosts by a fully connected attributed graph-$G=(V, E)$, where V is the set of nodes, with each infected host standing for a node, and E is the set of edges. A node is added to the set V when a new host exhibits malicious activity. On the other hand, a node is removed from V when the corresponding host ceases to exhibit malicious activity and is reliably known to have been disinfected. These nodes can be added back to V if and when the corresponding hosts exhibit malicious activity again.

What makes the graph G attributed is that the edges have weights associated with them. The weight of an edge connecting two nodes is the likelihood that the two nodes are part of the same botnet. Edge weights take values between 0 and 1. An edge weight close to 1 indicates that there is a high probability that the two nodes are part of the same botnet, whereas an edge weight close to 0 indicates that it is almost certain that the two nodes belong to different botnets. Edge weights are updated at the end of each time slot to reflect the observations in that particular time. For instance, if the two nodes performed the same malicious activity during this period, the corresponding edge weight is increased because our confidence that the two nodes are part of the same botnet will have increased. Weights for different activities differ and are theoretically derived (in a manner we will explain shortly) to represent the confidence we can place in our observation of those activities. As hosts belonging to the same botnet tend to exhibit temporally similar activities over an extended time, the edges between such hosts in graph G acquire large weights. On the other hand, the edges between the hosts that belong to two different botnets exhibit low edge weights.

Assuming that a network of computers has been attacked by multiple botnets, for botnet detection and identification, we are interested in partitioning G into disjoint subgraphs, with each subgraph corresponding to a unique botnet. This obviously assumes that each infected machine can be assumed to be under the control of a single botnet at any given time—an assumption whose justification was provided in Section 3. We want our botnet detection algorithm to assign each infected node a unique botnet label. We use the *Normalized Cuts* algorithm developed in [27] to segment the graph. Graph partitioning can be performed repeatedly to assign botnet labels as new bots/botnets make appearances in the network.

Figure 5 summarizes the different stages of our approach.

4.1. Malicious activity detection at a host

To detect and measure the activity co-occurrences needed by our approach, one must have in place the tools needed for the detection of malicious activities. Over the years, various methods have been developed for that purpose. For example, the router in charge of a local area network can be equipped with a monitor that can log network traces associated with each client in its Dynamic Host Configuration Protocol (DHCP) and static-IP-address client lists. Such logging of network traces is supported by many routers, such as those made by Cisco and Juniper. Subsequently, these logs may be analyzed to detect if malicious activities have occurred at the hosts.

To cite some previous efforts that perform efficient detection of malicious activities, scanning activities can be detected by monitoring the failed connection rate from a host as in SCADE [21]. Spamming by a host can be detected by monitoring the following: (i) the frequency of Domain Name System queries for mail exchanger records emanating from the host; and (ii) the frequency at which the host initiates the Simple Mail Transfer Protocol connections to outside mail servers [1]. Malicious download detection can be carried out by using Snort's signature detection engine. Other malicious activities can be similarly detected using freely available network intrusion detection and prevention systems. Alternatively, such malicious activities can be detected locally at a host by using tools such as Snort and BotHunter.

Our contribution in this paper is not to rebuild tools such as those mentioned earlier. Rather, our aim is to use the output of the tools already in existence to detect malicious activities and to then use the measures of those activities to detect and identify the botnets by using a graph-based statistical framework for the analysis of data.

4.2. Updating edge weights

Assuming that the activities listed in Table I have been measured during discretized time intervals, we will now explain how the initial graph G comes into existence and how its edge weights are subsequently updated as the activity measures change with time. We will assume that each discretized time interval is of unit duration.

Let i, j be two nodes in graph G, that is, these are hosts in the network that have exhibited signs of malicious activity. Let C_{ij} be a (latent) random variable indicating if the hosts i, j are part of the same botnet or not: $C_{ij} = 1$ indicates that the two hosts are part of the same botnet, and $C_{ij} = 0$ indicates that they are not. We are interested in computing $p(C_{ij} = 1)$ to estimate the probability that the two hosts i, j are part of the same botnet on the basis of the observations up to the current time t. At any given time t, the edge weight between the two nodes i, j is denoted as $p(C_{ij}^t = 1)$. Because the edge weight is a probability, it takes on a value between 0 and 1. As time is discretized into unit–duration intervals, we have $t = \{0, 1, 2, \ldots\}$. The edge update at each time t will be based on the observations of malicious activity between $t-1$ and t.

Let $A_i^{(t)}$ and $A_j^{(t)}$ be the random variables that describe the activity shown by the ith and jth host, respectively, between times $t-1$ and t. Let a_i and a_j be the actual values taken on by the variables $A_i^{(t)}$ and $A_j^{(t)}$. a_i and a_j then index into the malicious activity table shown in Table I. a_i can take values from 0, 1, …, K, where $a_i = 0$ indicates that the host showed no malicious activity during the timeslot under consideration, whereas $a_i = m$ $(1 \le m \le K)$ indicates that the host performed the mth malicious activity in Table I between $t-1$ and t (because there are seven malicious activities listed in Table I, $K = 7$).

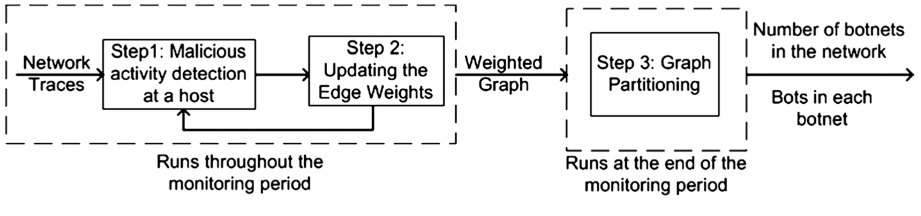

Network Traces
Step1: Malicious activity detection at a host
Step 2: Updating the Edge Weights
Weighted Graph
Step 3: Graph Partitioning
Number of botnets in the network
Bots in each botnet
Runs throughout the monitoring period
Runs at the end of the monitoring period

Figure 5. Stages in our approach.

At each discrete time t, we update the prior value of $p(C_{ij}^{(t-1)})$ between every pair of nodes to a posterior value $p(C_{ij}^{(t)})$ in a probabilistic fashion by using Bayes' rule. The posterior (updated) edge weight between two nodes depends on the previous edge weight $p(C_{ij}^{(t-1)})$, the likelihood of seeing this group of activities ($A_i^{(t)}$ and $A_j^{(t)}$) given the value of C_{ij}, and the overall likelihood of seeing this pair of activities $p(A_i^{(t)}, A_j^{(t)})$:

$$p\left(C_{ij}^{(t)}\middle|A_i^{(t)} = a_i, A_j^{(t)} = a_j\right) = \frac{p\left(A_i^{(t)} = a_i, A_j^{(t)} = a_j\middle|C_{ij}^{(t)}\right)p\left(C_{ij}^{(t-1)}\right)}{p\left(A_i^{(t)} = a_i, A_j^{(t)} = a_j\right)} \tag{1}$$

The likelihoods $p(A_i^{(t)} = a_i, A_j^{(t)} = a_j|C_{ij} = 1)$ and $p(A_i^{(t)} = a_i, A_j^{(t)} = a_j|C_{ij} = 0)$, which are essential for calculating the left-hand side of Equation (1), are estimated for every pair of the possible botnet activities listed in Table I. These joint probabilities are estimated theoretically, as discussed in the following section. It is important to realize that there is never a need to directly estimate the denominator on the right-hand side of Equation (1), that is, because the normalization constraint on the probabilities associated with the two outcomes for C_{ij} indirectly yields the denominator in Equation (1). That is, if we estimate these two probabilities without knowledge of the denominator, then the fact that these probabilities must add up to 1 will give us the unknown value for the denominator.

Edges are updated between a pair of hosts i, j at the end of a timeslot only if at least one of them exhibited malicious activity during the timeslot, that is, either $A_i^{(t)} \neq 0$ or $A_j^{(t)} \neq 0$.

At the start of the monitoring period, that is, when the graph G is first constructed, the edge weights are initialized to 0.5 between every pair of nodes exhibiting malicious activity. A weight of 0.5 indicates that the odds are even that the hosts at the two endpoints of the edge are infected by the same botnet, on the one hand, and by two different botnets, on the other hand. If a network is infected by only one or two botnets, one could argue for this initial probability value to be higher. But then, one can also argue for this initial probability to be lower if a network has been attacked by many small botnets. In the absence of any prior information regarding the number and/or the sizes of the botnets in a network, we choose to make an unbiased initialization of 0.5 for all the edges between the nodes exhibiting activities of the sort shown in Table I. After the graph G is created, when a new host in the network starts showing signs of malicious activity (and is not part of G yet), it is added to G, and new edges are created from the just infected host to each node already in G with an initial unbiased edge weight of 0.5.

4.2.1. Estimating the joint activity distributions.

As mentioned earlier, $A_i^{(t)}$ and $A_j^{(t)}$ are the activities detected by the network monitor at the nodes i and j, respectively, between the instants $t-1$ and t. We are interested in estimating the joint activity distributions $p(A_i^{(t)} = a_i, A_j^{(t)} = a_j|C_{ij} = 1)$ and $p(A_i^{(t)} = a_i, A_j^{(t)} = a_j|C_{ij} = 0)$ for all possible pairings of a_i and a_j.

Let $B_i^{(t)}$ and $B_j^{(t)}$ be the botmaster-ordered activities at the nodes i and j between times $t-1$ and t, with b_i and b_j being two specific values for such activities. Just similar to $A_i^{(t)}$, $B_i^{(t)}$ is a $K+1$-valued random variable, with K being the number of activities in Table I. Note that $b_i = 0$ indicates that the botmaster chose no malicious activity for the timeslot (i.e., let his botnet idle) and that $b_i = l$ $(1 \leq l \leq K)$ indicates that the botmaster picked the lth malicious activity in Table I to be performed between $t-1$ and t. The probabilities $p(B_i^{(t)} = b)$ represent the relative frequencies of the different malicious activities ordered by the botmaster. On the basis of the empirical data, we can assume that, on average, a botmaster lets his botnet idle 25% of the time. For the rest of the time, the activities ordered by a botmaster are according to the relative frequencies in Table I.

The specific values taken on by A_i and A_j at time t depend on the activities B_i and B_j selected by the botmasters of the nodes i and j for that time slot. We must also allow for the possibility that both nodes might be controlled by the same botmaster.

Recall the use of C_{ij} to represent whether or not the two bots i and j are part of the same botnet. If $C_{ij} = 1$, that indicates that i and j are part of the same botnet and have the same botmaster, and when $C_{ij} = 0$, the bots i and j are under the control of two different botmasters. We are obviously interested in the probability that two different nodes exhibit two given activities when both nodes are in the same botnet and when they are not. The following formula expresses these joint probabilities:

We will now show simpler versions of the aforementioned formula for the two cases of $C_{ij} = 1$ and $C_{ij} = 0$.

$$\begin{aligned} p(A_i^{(t)} = a_i, A_j^{(t)} = a_j\big|C_{ij}) &= \sum_{b_i=0}^{K}\sum_{b_j=0}^{K} p(A_i^{(t)} = a_i, A_j^{(t)} = a_j\big|B_i^{(t)} = b_i, B_j^{(t)} = b_j, C_{ij}) \\ &\quad \times p(B_i^{(t)} = b_i, B_j^{(t)} = b_j\big|C_{ij}) \end{aligned} \tag{2}$$

When $C_{ij} = 1$, both the bots i and j are controlled by the same botmaster. We indicate that fact by saying $B_i = B_j = B$, that is, B is now the random variable for the common activity at the two nodes. In this case, Equation (2) becomes

$$p(A_j^{(t)} = a_i, A_j^{(t)} = a_j \Big| C_{ij} = 1) = \sum_{b=0}^{K} p(A_i^{(t)} = a_i, A_j^{(t)} = a_j \Big| B^{(t)} = b) \times p(B^{(t)} = b) \quad (3)$$

where b is any specific value for the random variable B, the common activity ordered by the botmaster for the botnet. Note that we have removed the dependence on C_{ij} in Equation (3) because the fact that the two bots have the same botmaster implicitly assumes that $C_{ij} = 1$. To further simplify the equation shown earlier, we note that $A_i^{(t)}$ is independent of $A_j^{(t)}$ because the activity actually carried out by bot i depends only on the activity selected by the bot's botmaster and is independent of what is going on at the bot j. We can, therefore, rewrite the aforementioned equation as

$$p(A_i^{(t)} = a_i, A_j^{(t)} = a_j | C_{ij} = 1) = \sum_{b=0}^{K} p(A_i^{(t)} = a_i | B^{(t)} = b) \times p(A_j^{(t)} = a_j | B^{(t)} = b) \times p(B^{(t)} = b) \quad (4)$$

In the other case, that is, when $C_{ij} = 0$, the bots i and j have two different botmasters B_i and B_j. Again, the dependence on $C_{ij} = 0$ can be removed in the formula for the joint probability $p(A_i^{(t)} = a_i, A_j^{(t)} = a_j | C_{ij} = 0)$ as long as we maintain the distinction between the activities B_i and B_j at the two nodes. As with the $C_{ij} = 1$ case, $A_i^{(t)}$ is again independent of $A_j^{(t)}$ because the activity actually carried out by bot i depends only on the activity selected by the bot's botmaster and is independent of what is going on at the bot j. We can, therefore, rewrite Equation (2) for the case $C_{ij} = 0$ as follows:

$$p(A_i^{(t)} = a_i, A_j^{(t)} = a_j \Big| C_{ij} = 0) = \sum_{b_i=0}^{K} p(A_i^{(t)} = a_i \Big| B_i^{(t)} = b_i)\ p(B_i^{(t)} = b_i) \times \sum_{b_j=0}^{K} p(A_j^{(t)} = a_j \Big| B_j^{(t)} = b_j)\ p(B_j^{(t)} = b_j) \quad (5)$$

Both Equations (4) and (5), which provide the joint distributions over the observed malicious activities at a pair of hosts under the condition that they belong to the same botnet and under the condition that they belong to two different botnets, depend on us being able to estimate $p(A^{(t)} = a|B^{(t)} = b)$—the probability that the observed malicious activity at a bot is a while the botmaster chose b. This probability can be ascertained from the following considerations:

- The probability $p(A^{(t)} = a|B^{(t)} = b)$ depends obviously on whether the infected host for which the probability is being estimated is in the *Offline* state as a bot or the *Online* state.
- When an infected host is in the *Online* state, we assume it can receive the commands of the botmaster and do the bidding of the botmaster as directed.
- We must also account for the possibility that a network monitor may mis-identify the malicious activity at a bot.

As was mentioned earlier in Section 3 and for reasons presented in that section, of all the hosts infected by a botnet, typically only a small number will be in the state *Online* at any given time. It has been observed empirically that the time a bot remains in the state *Online* is exponentially distributed with mean α, whereas the time it is in the state *Offline* is also exponentially distributed but with mean β. On the basis of experimental observations on botnets, α and β are believed to be roughly 25 and 50 min, respectively [6], so we can say that the probability that an infected host is in the bot state *Online* at any given instant is

$$p_{\text{on}} = \frac{\alpha}{\alpha + \beta} \quad (6)$$

Let us now talk about the probability of a network monitor mis-identifying the malicious activity at a node. We will denote this probability by p_{error}. If $\hat{A}_i^{(t)}$ is the true activity performed by bot i between $t-1$ and t and $A_i^{(t)}$ is the activity as detected by the network monitor, then obviously, $p_{\text{error}} = p(A_i^{(t)} \neq \hat{A}_i^{(t)})$. We assume that this probability of erroneous detection by the network monitor is uniformly distributed over all possible ways to mis-identify a given malicious activity.

We now bring together these considerations into the following result for the probability $p(A^{(t)} = a|B^{(t)} = b)$:

$$p(A^{(t)} = a \big| B^{(t)} = b) = p_{\text{on}}\ [(1 - p_{\text{error}})\ 1(a = b) + \frac{p_{\text{error}}}{K}\ 1(a \neq b)] + (1 - p_{\text{on}})\ 1(a = 0) \quad (7)$$

where $1(s)$ is an indicator function that equals 1 when its argument s is true and 0 otherwise. To understand the right-hand side earlier, note that, in the absence of observation errors (which happens with probability $1 - p_{\text{error}}$), the node where a is observed must exhibit the same activity as the activity b ordered by the botmaster provided, of course, the node is in the state *Online*, which happens with probability p_{on}. To this, we must add the probability of the monitor erroneously telling us that the activity a was observed while the true activity at the node was really b—the activity ordered by the botmaster. The last part of the right-hand side earlier is the estimate of $p(A^{(t)} = a|B^{(t)} = b)$ for the special case when $a = 0$, meaning when the infected host to which $A^{(t)}$ applies is in the state *Offline*, which obviously happens with the probability $(1-p_{\text{on}})$.

Using the formulas shown earlier, the estimated joint distributions $p\left(A_i^{(t)}, A_j^{(t)} \middle| C_{ij}=1\right)$ and $p\left(A_i^{(t)}, A_j^{(t)} \middle| C_{ij}=0\right)$ are given in Tables II and III, respectively. For the estimates shown in the tables, we assumed $p_{\text{error}}=1\%$, $\alpha=25$, and $\beta=50$.

The equations provided in this section make relatively few assumptions and are based on the observed behavior of the botnets. We will use them to estimate the temporal activity co-occurrences at pairs of infected nodes under the condition that both nodes in a pair are infected by the same botnet and under the condition that the infection at the two nodes is by two different botnets.

These activity distributions will be used to update the edge weights in the graph representation of the infected hosts in a network, as discussed earlier, at the end of each unit of observation time. At the end of the monitoring period, we will end up with a graph representation in which the edge weights give us the likelihood that the corresponding nodes belong to the same botnet. We, then, segment out the botnets in the graph by using graph partitioning, the details of which are discussed in the succeeding text.

4.3. Graph partitioning

The input to the graph-partitioning step is a graph whose nodes are the hosts that have exhibited malicious activities of the sort listed in Table I and whose edges represent the likelihood that the two nodes linked by an edge are part of the same botnet. The likelihood of any two nodes belonging to the same botnet is based on the cumulative temporal co-occurrences in the malicious activities exhibited at those nodes.

Only those hosts whose activity counts exceed a certain threshold are included in the graph to be partitioned. This is to ensure that the nodes to be used for botnet identification, and which are to be assigned a botnet label, have shown a more or less stable pattern of malicious behavior. A count of 15 for the threshold has worked well in our experiments. By that, we mean that a node is included in the graph to be partitioned if the monitor has reported at least 15 malicious activity counts at the node, which could be 15 of the same activity over 15 units of observation time or a total of different malicious activities over whatever period it takes for the count to add up to 15.

We are now interested in segmenting G into different botnets, assuming that more than one botnet has infected the network—as is not uncommonly the case. We will use the *Normalized Cuts* algorithm [27] for the same. This algorithm involves the following steps:

- Estimate a $|V|$ dimensional partition vector to bi-partition the graph.
- Decide if the partitions created should be further subdivided, and repartition the segmented parts if necessary.

4.3.1. Bi-partitioning the graph.

Given the graph $G=(V, E)$, we are interested in partitioning it into two disjoint subgraphs, with one involving the subset P of the nodes and the other the subset Q of nodes such that $P \cup Q = V$ and $P \cap Q = \varnothing$. The cost of bi-partitioning is the weight of the edges that go from the nodes in one partition to the nodes in the other partition.

Table II. $p\left(A_i^{(t)}, A_j^{(t)} \middle| C_{ij}=1\right)$.

Activity	0	1	2	3	4	5	6	7
0	0.5855	0.054489	0.046287	0.024961	0.011838	0.011838	0.011838	0.0052759
1	0.054489	0.026417	7.0583e-05	5.5588e-05	4.636e-005	4.636e-005	4.636e-005	4.1746e-005
2	0.046287	7.0583e-005	0.022414	4.9821e-005	4.0593e-005	4.0593e-005	4.0593e-005	3.5979e-005
3	0.024961	5.5588e-005	4.9821e-005	0.012008	2.5598e-005	2.5598e-005	2.5598e-005	2.0984e-005
4	0.011838	4.636e-005	4.0593e-005	2.5598e-005	0.0056037	1.6371e-005	1.6371e-005	1.1757e-005
5	0.011838	4.636e-005	4.0593e-005	2.5598e-005	1.6371e-005	0.0056037	1.6371e-005	1.1757e-005
6	0.011838	4.636e-005	4.0593e-005	2.5598e-005	1.6371e-005	1.6371e-005	0.0056037	1.1757e-005
7	0.0052759	4.1746e-005	3.5979e-005	2.0984e-005	1.1757e-005	1.1757e-005	1.1757e-005	0.0024017

Table III. $p\left(A_i^{(t)}, A_j^{(t)} \middle| C_{ij}=0\right)$.

Activity	0	1	2	3	4	5	6	7
0	0.56555	0.061074	0.051875	0.027955	0.013235	0.013235	0.013235	0.0058745
1	0.061074	0.0065956	0.005602	0.0030189	0.0014292	0.0014292	0.0014292	0.0006344
2	0.051875	0.005602	0.0047582	0.0025641	0.0012139	0.0012139	0.0012139	0.00053884
3	0.027955	0.0030189	0.0025641	0.0013818	0.00065417	0.00065417	0.00065417	0.00029037
4	0.013235	0.0014292	0.0012139	0.00065417	0.0003097	0.0003097	0.0003097	0.00013747
5	0.013235	0.0014292	0.0012139	0.00065417	0.0003097	0.0003097	0.0003097	0.00013747
6	0.013235	0.0014292	0.0012139	0.00065417	0.0003097	0.0003097	0.0003097	0.00013747
7	0.0058745	0.0006344	0.00053884	0.00029037	0.00013747	0.00013747	0.00013747	6.1021e-005

For optimal bi-partitioning, we would want to determine P and Q so that this cost is minimized. The following formula is a normalized version of this cost:

$$\mathrm{Ncut}(P, Q) = \frac{\mathrm{cut}(P, Q)}{\mathrm{assoc}(P, V)} + \frac{\mathrm{cut}(P, Q)}{\mathrm{assoc}(Q, V)} \tag{8}$$

where

$$\mathrm{cut}(P, Q) = \sum_{i \in P,\, j \in Q} w_{ij} \tag{9}$$

and

$$\mathrm{assoc}(P, V) = \sum_{i \in P,\, j \in V} w_{ij} \tag{10}$$

where w_{ij} is the edge weight between the nodes i and j. In our case, $w_{ij} = p\left(C_{ij}^{(t)}\right)$. Minimizing the normalized cost $Ncut(P, Q)$ simultaneously minimizes the association between the partitions and maximizes the association within a partition. This creates tight, well separated partitions.

As explained in [27], we seek a $|V|$ dimensional partition vector $\mathbf{y}$ that minimizes the *Ncut* cost function such that $y_i = +2$ if node i is in P and $-2b$ if the node is in Q, where b is the ratio of the sum of the weighted vertex degrees in the two subgraphs yielded by the partition. For our discussion here, the important point is that $\mathbf{y}$ is a vector with binary valued elements. Unfortunately, determining such a vector $\mathbf{y}$ directly is known to be NP-Hard. However, an approximate discrete solution can be obtained by allowing the elements of $\mathbf{y}$ to take on real values. Subsequently, the calculated value for the elements of $\mathbf{y}$ can be thresholded to yield a good approximation to the optimal solution.

Let W be an $N \times N$ symmetric matrix where $W(i, j) = w_{ij}$. Let $d(i) = \sum_j w_{ij}$ where $d(i)$ measures the total connection weight from node i to all other nodes. Let D be an $N \times N$ diagonal matrix with d on its diagonal. If $\mathbf{y}$ is relaxed in the manner described earlier, we can obtain a good approximation to the optimal solution for the *Ncut* problem by solving the following generalized eigendecomposition problem:

$$(D - W)\mathbf{y} = \lambda D\mathbf{y} \tag{11}$$

We can transform Equation (11) to a standard eigen system of equations by writing it as

$$D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}\mathbf{z} = \lambda\mathbf{z} \tag{12}$$

where $\mathbf{y} = D^{-\frac{1}{2}}\mathbf{z}$. The second smallest eigenvector, represented by $\mathbf{z}_1$, is then the solution we want for the aforementioned system of equations. This solution can be turned into a solution for $\mathbf{y}$ by $\mathbf{y} = D^{-\frac{1}{2}}\mathbf{z}_1$. The continuous values obtained for $\mathbf{y}$ as explained earlier can subsequently be discretized by using a threshold that achieves the lowest *Ncut* value.

After we split the graph G into two parts by using the vector $\mathbf{y}$, we determine if the partitions created should be further subdivided. If it is deemed that further partitioning is to take place, then each partition is subdivided by following the steps outlined earlier, else the recursion for that partition terminates.

4.3.2. Recursive partitioning.

A recursive application of the algorithm we have described so far is used to further partition the graph as long as the following two criteria are met: (i) the partition vector is stable; and (ii) the *Ncut* value of a partition vector is below a certain threshold, as explained in [27]. With regard to the stability of the partition vector, we do not want partition vectors whose element values are smoothly varying (such partition vectors are ambiguous because they exhibit multiple possibilities for the splitting point). The possibility that a given $\mathbf{y}$ consists of smoothly varying values can be ascertained by making a histogram of the values of the vector elements and by computing the ratio between the minimum and maximum values in the bins. The ratio will be low for stable partition vectors. In our experimental work, we reject partitions for which this ratio, defined as the *Cut Stability Threshold*, is greater than 0.7.

The second criterion is based on the *Ncut* value of the partition vector. We would want partition vectors to have an *Ncut* value as low as possible. Hence, we must reject partition vectors with *Ncut* values above a threshold—the *Ncut* threshold. This threshold is based on how discriminating one wants to be with regard to separating out the botnets that may behave very similarly.

5. A UNIFIED BOTNET MODEL FOR VALIDATING DETECTION FORMALISMS

An important issue related to any new formalism for botnet detection is its validation. Validation is particularly challenging for formalisms that seek to take into account the temporal aspects of botnet activities, activity durations, bot lifetimes, botnet growth and decay, and so on. Because any statistically meaningful validation must of necessity involve a large network, a controlled injection of botnets into such networks, placement of bot monitors at each of the computers in the network, and so on, it is easy to see why truly experimental validations are not feasible in real life. One must therefore resort to the alternative of simulating botnets that take into account our latest collective understanding of botnet behaviors and lifetimes. The goal of this section is to present a unified model of botnets that pulls together the latest understanding in the research community about how botnets behave, bot lifetimes, and the growth and decay of botnets.

The unified model consists of two main components—a behavioral component that captures the activities that a botnet carries out, how long the bots of a botnet stay online, and so on (these are based on the observations in [6,11]), and a propagation component that captures how a botnet grows and decays over time. The propagation of botnets is based on virus infection models proposed in [9,14], and [15]. Because botnets are essentially viruses/worms that are centrally controlled, these virus propagation models are readily extended to modeling the spread of botnets.

Our model uses discrete time. Let N be the number of hosts in a network. At any given time t, each host in the network is in one of the three categories: ***Infected* (*I*), *Susceptible* (*S*), or *Cured* (*C*)** (based on the categorization in [9]). Hosts in the *Infected* category are part of some botnet, and each host has a unique botnet label. Hosts that are in the *Susceptible* category are presently not infected but can be infected in the future. Hosts in the *Cured* category are machines that were recently disinfected and are immune to new infection attempts for a certain (fixed) time. After their immunity to infections expires, these hosts will move from the *Cured* category to the *Susceptible* category. The number of hosts in each category varies with time as some *Susceptible* hosts may become infected, whereas some existing *Infected* hosts may become cured. At any time t, the hosts in all the categories sum to N (the total number of hosts in the network), that is, as follows:

$$I(t) + S(t) + C(t) = N \tag{13}$$

We are interested in modeling the hosts in the *Infected* category as these are the computers in the network that are members of some botnet. As discussed earlier, at any given time, each such bot is in one of the two states *Online* or *Offline*. The time durations for which a bot stays *Online* or *Offline* are exponentially distributed with means α and β, respectively. This model of bot behavior is based on observations in [6].

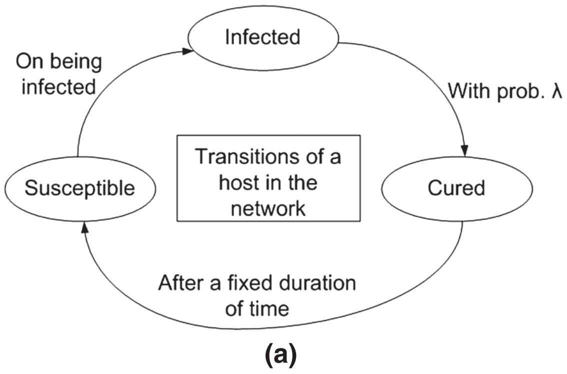
At time t, the hosts that exhibit malicious behavior are those that are part of the *Infected* category and are in the bot state *Online*. These are the hosts that are active bots and follow the commands dictated by the botmaster. As discussed earlier, botmasters occasionally let their botnets idle. During this time, even those bots of the botnet that are *Online* will not perform any malicious activity. For the remaining time, botmasters command their botnets to perform malicious activities shown in Table I (with associated frequencies). The duration (in min) of each malicious activity chosen is exponentially distributed with mean γ. The malicious activities (or the inactivity) chosen by the botmasters of the different botnets at a specific time are independent of one another, as are the durations for which the activities are executed.

One of the activities that can be chosen by a botmaster is the scanning activity. This is the activity where bots of a botnet scan other computers to infect them. When the botmaster of a botnet chooses the scanning activity, each bot of that botnet that is *Online* sends an infection to the other hosts in the network with probability ρ. Hosts that are sent an infection by the scanning hosts can belong to any category (*Infected*, *Cured*, or *Susceptible*). Infection attempts on hosts that are in the *Infected* or the *Cured* category have no effect, that is, the hosts in the *Infected* category maintain their botnet label, whereas the hosts in *Cured* category remain immune to infection attempts. Infection attempts only succeed on hosts in the *Susceptible* category. These hosts are then assigned the same botnet label as the botnet label of the host that infected them and move into the *Infected* category at the next sampling instant. Their behavior is then governed by the bot model described earlier, where they transition between *Online* and *Offline* states and obey the commands of the botmaster while *Online*.

Finally, the hosts in the *Infected* category can become cured at any time with a probability λ. If a host becomes cured, it moves from the *Infected* category to the *Cured* category.

Figure 6 shows the transitions of hosts between the different states in a network.

Table IV summarizes the symbols in the model.


Infected
On being infected
With prob. λ
Transitions of a host in the network
Susceptible
Cured
After a fixed duration of time
(a)

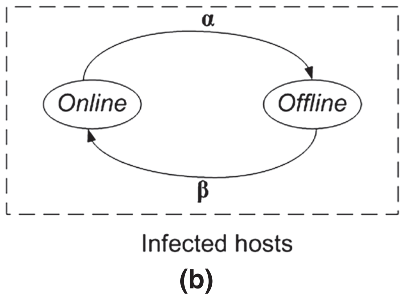


Figure 6. (a) The transitions of a host in a network between the different states related to infection in the botnet model, (b) transition of an infected host between the bot states *Online* and *Offline*.

Table IV. Symbols in botnet model.

Symbol	Stands for
N	Number of hosts in a network
B	Number of botnets in the network
t	The discrete timeslot under consideration
$I(t)$	Number of Infected hosts at time t
$S(t)$	Number of Susceptible hosts at time t
$C(t)$	Number of Cured hosts at time t
α	Mean time a bot stays *Online*
β	Mean time a bot stays *Offline*
γ	Mean time a malicious activity is performed
ρ	The probability of infecting a new host
λ	The probability of an infected host becoming disinfected

6. EXPERIMENTS AND RESULTS

Our goal in this section is to provide experimental and simulated validation for the proposed framework for botnet detection. We will show results on two real network traffic traces and one simulated trace that is based on the new unified botnet model presented in Section 5.

6.1. Experiment 1

The first network trace that we show our results on consists of a monitored network of 15 hosts. There are two IRC botnets in the network, each with three hosts. Hosts labeled 1 through 3 are part of botnet 1 and those labeled 4 through 6 are part of botnet 2. All the computers are involved in regular Internet communication with activities including P2P downloads, HTTP web communication, File Transfer Protocol, Telnet, and so on. The malicious hosts are involved in C&C communications with the IRC server, DDoS attacks, and port scans. There is significant overlap in time when both botnets are involved in the same activities. Additionally, there are no unique activities that can be used to segment out the individual botnets.

We monitored the network continuously for 24 h, during which time we detected malicious activities. Port scans and attacks being performed by the malicious hosts in the network were detected using Snort and patterns of failed connection rates. During this time, we continuously update the edges between the nodes in the graph (each node corresponds to a malicious host in the network) by using Equation (1). The resulting edge weights between the nodes (corresponding to hosts 1 through 6 in the network) are shown in Table V. As can be seen in Table V, the edge weights between the nodes belonging to the same botnet are high, whereas the edge weights between the nodes belonging to different botnets are low. For instance, the edge weight between the nodes 1 and 3 (which belong to the same botnet) is 0.94, whereas the edge weight between the nodes 1 and 4 (which belong to different botnets) is 7×10^{-3}.

This updated edge weight matrix is then partitioned to identify the botnets in the network. Table VI shows the recursive steps in the graph-partitioning algorithm. Both the *Ncut* and the *Cut Stability* thresholds were set to 0.7. The graph is recursively partitioned as long as the *Ncut* value of the new partition is lesser than the set *Ncut* threshold and the cut is stable (it achieves a stability value lower than the *Cut Stability* threshold). The values of the two criteria that do not meet the thresholds are underlined.

For this real network trace, our approach correctly identifies both the bots in each botnet and the number of botnets in the network.

6.2. Experiment 2

We validate the performance of our approach on the botnet datasets generated on the basis of the botnet model discussed in Section 5. We test our botnet detection approach in a wide variety of situations—differing numbers of botnets in the network, varying activity levels, and varying growth patterns of botnets. All the experiments were run on a network of 200 hosts.

The variations in our datasets are presented in the following text.

6.2.1. Variation in the number of hosts in different categories.

Experiments were conducted in which the number of hosts in different categories (*Infected*, *Susceptible*, and *Cured*) exhibit different patterns over the course of a simulation, such as those shown in 7a and 7b. These patterns depend on the number of initial infected hosts ($I(0)$) and the number of botnets in the network. The value of $I(0)$ was varied from 30 to 108. The number of botnets in the networks also plays a major role in this experiment. In Figure 7a, there are 20 botnets in the network that all compete for the susceptible hosts; as a result of which, the number of infected hosts remains high. On the other hand, Figure 7b with just three botnets shows a greater availability of susceptible hosts.

6.2.2. Variation in the number of botnets.

The initial number of botnets in the network (B in Table IV) is varied—simulations are repeated with 3, 5, 9, and 20 initial botnets.

Having differing numbers of initial botnets creates different types of conditions for the graph-partitioning algorithm for botnet detection. When there are a small number of botnets, say just three, it is less likely that any botnet will be completely eliminated from the network—because botnets will be able to acquire new bots at a rate that is comparable with the rate at which bots of the botnet become cured. However, having a large number of botnets, for example 20, would lead to the elimination of some botnets. Figure 8 shows this variation as observed in our datasets with 20, 9, and 3 initial botnets. We are particularly

Table V. Edge weights between malicious nodes in Experiment 1.

Node #	1	2	3	4	5	6
1	1.000	0.9202	0.9472	0.007	0.0007	0.0002
2	0.9202	1.000	0.968	0.0013	0.0001	0.0003
3	0.9472	0.968	1.000	0.0823	0.0006	0.0003
4	0.007	0.0013	0.0823	1.000	0.9714	0.9575
5	0.0007	0.0001	0.0006	0.9714	1.000	0.9581
6	0.0002	0.0003	0.0003	0.9575	0.9581	1.000

Table VI. Steps in graph partitioning.

Nodes to partition	{1, 2, 3, 4, 5, 6}	{1, 2, 3}	{4, 5, 6}
Partition 1	{1, 2, 3}	–	–
Partition 2	{4,5,6}	–	–
Ncut value	0.029	$\underline{0.970}$	$\underline{0.958}$
Stability value	0.333	$\underline{1}$	0.5

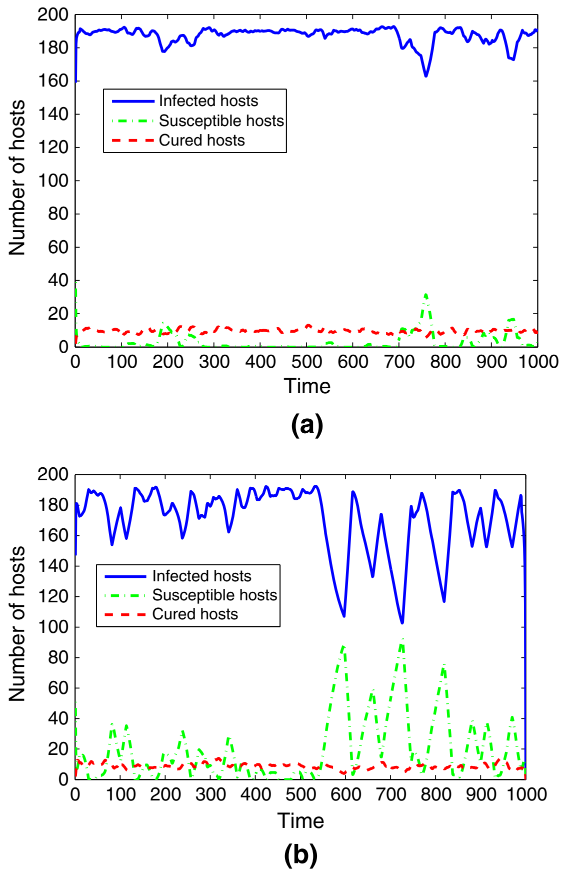


Figure 7. Variation of hosts in different categories (a) for $I(0) = 80$ and $B = 20$, and (b) for $I(0) = 108$ and $B = 3$.

interested in the variation of the number of botnets in the network, as it can be directly used to measure the accuracy of our graph-partitioning algorithm.

Additionally, as was shown in Figure 7, having differing numbers of botnets in simulations also causes the number of hosts in the different categories to exhibit varying patterns.

6.2.3. Variation in botnet sizes.

Botnet sizes vary rapidly over time—as new hosts are infected and existing infected hosts become disinfected. Figure 9 shows the variation in the sizes of four botnets in a simulation with 20 initial botnets. This rapid variation is a challenge for botnet detection, as new infected hosts have to be added to the graph, activity co-occurrences have to be established, and accurate botnet labels have to be determined, while older infected hosts that have since been cured are removed.

6.2.4. Variation in online footprint.

Finally, we also varied the amount of time a bot stays in the *Online* state to measure the impact on detection performance. We varied p_{on} from 0.1 to 0.9, that is, from a bot being active 10%–90% of the time. When bots are *Online*

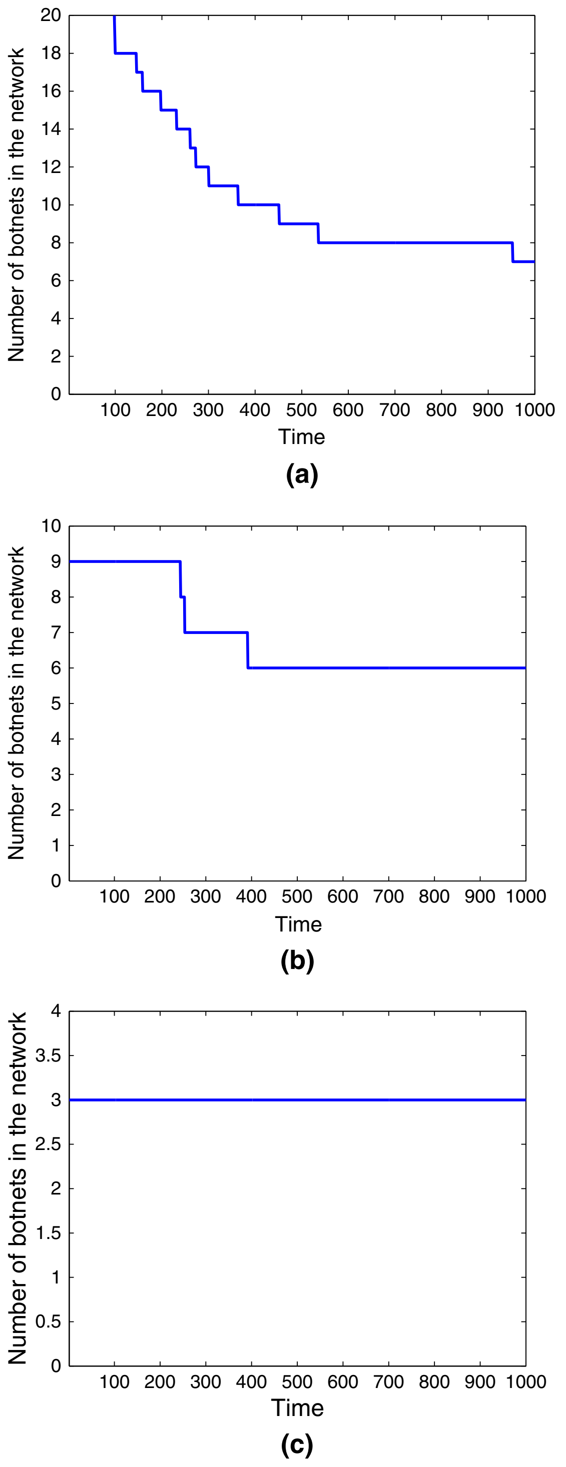


Figure 8. Variation in the number of botnets in the network over time for (a) 20 initial botnets, (b) 9 initial botnets, and (c) 3 initial botnets.

only briefly it becomes more difficult to establish strong temporal co-occurrences between the activities at the different hosts. Additionally, it is tougher to segment out

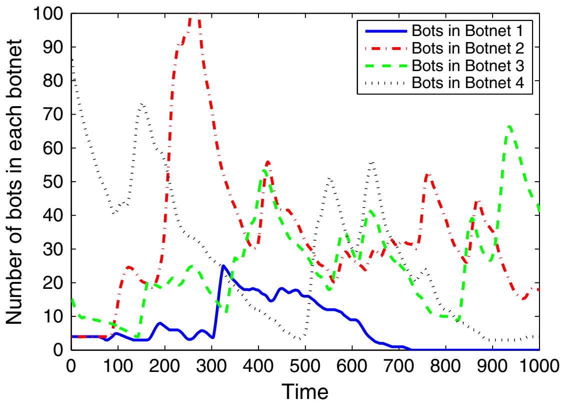


Figure 9. Variation in the number of bots in four botnets in a simulation with 20 initial botnets, $I(0) = 108$, $p_{on} = 0.5$. Botnet 1 (from the legend) is shown getting eliminated as all its bots have been cured, and it has been unable to obtain new bots in the meantime.

the complete botnet as different subsets may be online at different points in time.

6.2.5. Detection accuracy.

Tables VII and VIII summarize the parameters used for our simulations. As shown in Table VII, simulations are run with varying numbers of initial botnets, differing numbers of initial infected hosts, and varying activity levels of bots in the botnet (controlled by p_{on}). Bots in each of the botnets independently transition between *Online* and *Offline* states, and the percentage of time they stay *Online* is controlled by the activity level. As in Figure 9, botnet sizes vary rapidly. Table VIII summarizes the parameters common to all the experiments.

Figure 10 shows the detection accuracy of our approach on the datasets generated from these simulations. We measure the accuracy of assigning the correct botnet label to each malicious host in the network by comparing the assigned botnet label with the ground truth. In case the number of botnets estimated by the graph-partitioning algorithm differs from the true number of botnets at the time, we classify the biggest chunk of a botnet as correctly classified, whereas the rest of the botnet is deemed mis-classified.

The detection accuracy of our approach in Figure 10 is compared for different activity levels of the bots with different numbers of botnets in the network. Each point is the average of 15 simulation runs, and results obtained for differing number of initial infected hosts are averaged.

Table VII. Parameters for Experiment 2.

Number of botnets	$I(0)$	p_{on}
3	30, 72, 108	0.1, 0.3, 0.5, 0.7, 0.9
5	35, 76, 108	0.1, 0.3, 0.5, 0.7, 0.9
9	40, 70, 108	0.1, 0.3, 0.5, 0.7, 0.9
20	40, 60, 108	0.1, 0.3, 0.5, 0.7, 0.9

Table VIII. Common parameters for Experiment 2.

Common parameters	Values
N	200
γ	10
ρ	0.1
λ	0.01
p_{error}	0.01

As seen in Figure 10, the average detection rate of our approach is close to 90% for activity levels above 10% (i.e., bots are active more than 10% of the time). The approach assigns the correct botnet label to 95% or more of the bots in the network for the values of p_{on} between 0.5 and 0.9. We believe that this is impressive, considering that the number of botnets is continually varying, the number of bots within each botnet are varying rapidly, different subsets are online at different points in time, and all the botnets are involved in activities derived from the same underlying botnet activity distribution. The standard deviation of our approach is less than 3% for over 15 runs of simulation for each point, indicating that the framework consistently produces the same results.

The performance of our graph-based framework increases steadily with the increase in the activity levels of the individual bots. This is to be expected because higher activity levels give us more information to work with. However, there is a dip in the performance for the case when there are 20 botnets and the bots are online 90% of the time. Such high levels of activity lead to a substantially increased detection of co-occurrences between the different botnets, to the extent that it is difficult to segment them out.

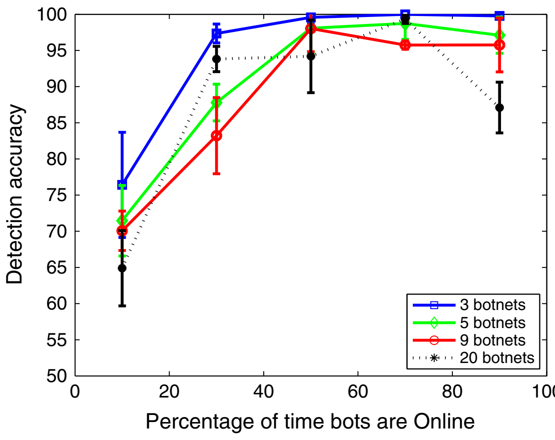
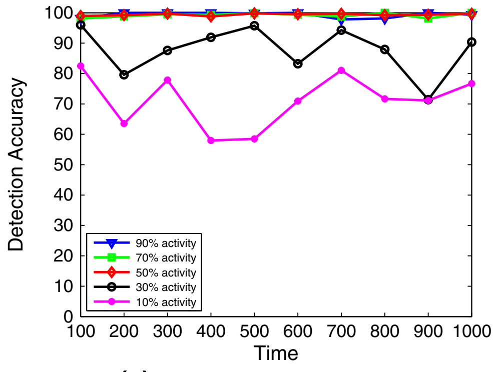
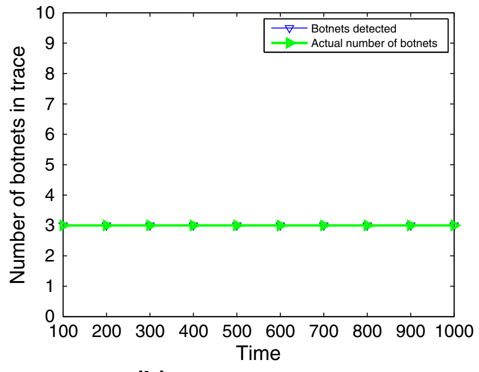


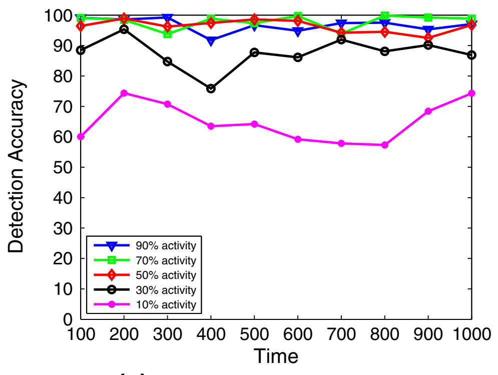
Figure 10. Detection accuracy for simulations with different numbers of initial botnets and varying activity levels of bots in the botnets.



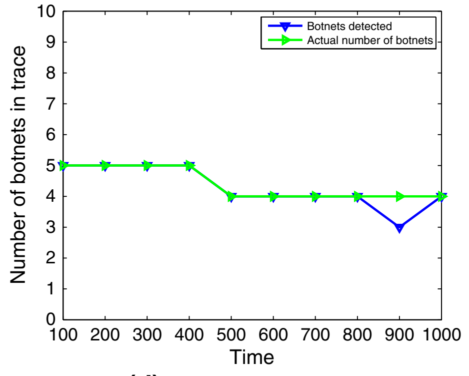
(a) Detection Accuracy: 3 botnets



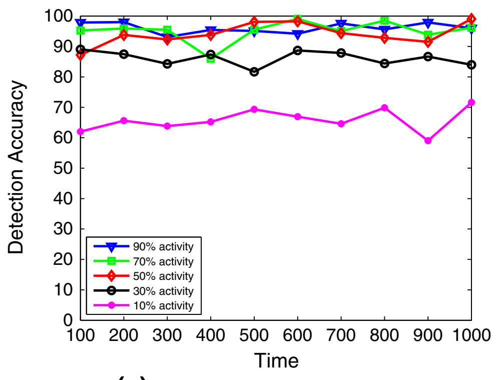
(b) 70% activity: 3 botnets



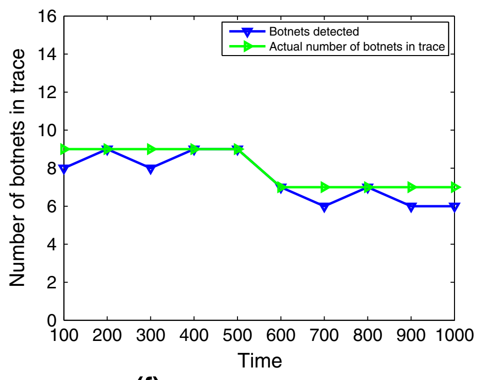
(c) Detection Accuracy: 5 botnets



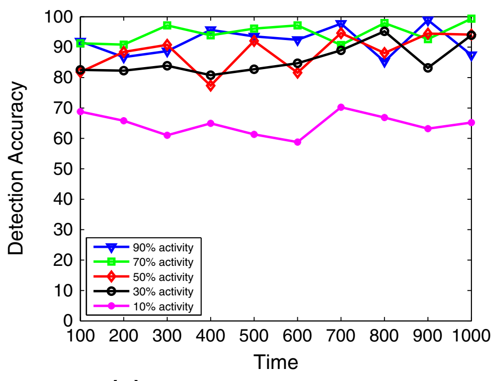
(d) 70% activity: 5 botnets



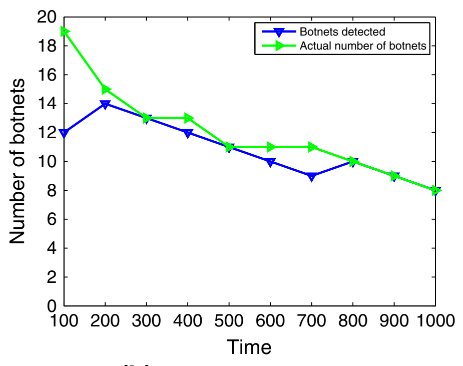
(e) Detection Accuracy: 9 botnets



(f) 70 % activity: 9 botnets



(g) Detection Accuracy: 20 botnets



(h) 70% activity: 20 botnets

Figure 11. (a, c, e, and g) Detection results of continuous monitoring on a network with different numbers of initial botnets. (b, d, f, and h) Comparison of the true versus the estimated number of botnets over time.

For the case of 10% activity, the average performance is around 72%. This reduced performance occurs because it is difficult to segment out complete botnets at low activity levels. At such activity levels, not only are the different bots online at different points in time but also each bot is online only very briefly, which leads to over segmentation of the botnet. Additionally, there is 1% erroneous detection, which makes accurate co-occurrences more difficult to establish. Given these constraints, these results are acceptable.

6.2.6. Continuous detection.

Figure 11 shows the results of using our graph-based framework for the case when a network is monitored continuously for the detection of botnets, and we have varying numbers of botnets and varying activity levels for each botnet. The initial number of infected hosts was 108 in each case. Note that the results shown are not averaged. That is, each result shown is obtained from a single experiment.

We detect and segment botnets every 100 units of time. As can be seen from the continuous detection results, the average performance is around 95% for activity levels between 50% and 90%, around 90% for activity levels of 30%, and around 70% for activity levels of 10%. We also show a comparison between the actual and the estimated number of botnets in the network for activity levels of 70% for all the experiments. The difference between them is never more than 1 at almost all instances, except at the first segmentation instance for the network of 20 botnets.

The proposed approach is efficient computationally. On an Intel 64-bit machine with 8 cores and 6 GB of random access memory, a single sequential network-wide edge update (i.e., the edge between every pair of hosts is updated once, and this process is repeated sequentially for all such edges) for a network of 200 hosts took only a couple of seconds. This is well under the duration of a timeslot. Recursive graph partitioning on the same sized network for 20 botnets took between 5 and 10 s. The algorithm was implemented with a mixture of Matlab, Perl, and C/C++.

6.3. Experiment 3

In our final experiment, we show the results of our approach on the infection profiles generated by a well-known bot detection tool called BotHunter, a tool developed and maintained by SRI International [21] (http://www.sri.com/). BotHunter detects the presence of a bot infection on a given host by passively monitoring the two-way communication between the host and the Internet. It then uses a set of predefined rules to declare a bot infection on the basis of different combinations of malicious activities seen. When enough evidence for reporting a bot infection has been gathered, an infection profile for the host is generated. Each infection profile contains information on several activities considered potentially malicious: Inbound Exploit, Egg Download, C&C Traffic, Outbound Scan, Outbound Attack, and so on.

Table IX. Datasets used for Experiment 3.

Dataset	No. of malicious hosts	No. of botnets	Bots in each botnet
1	39	5	4, 2, 23, 6, 4
2	61	5	2, 19, 33, 3, 4
3	51	7	6, 8, 5, 7, 2, 3, 20

For our experiment, we used three different sets of infection profiles, each generated over a period of 24 h. On the basis of the activity co-occurrences in time, we manually grouped the bots involved into constituent botnets. Table IX describes the number of botnets and bots within each botnet in the three datasets.

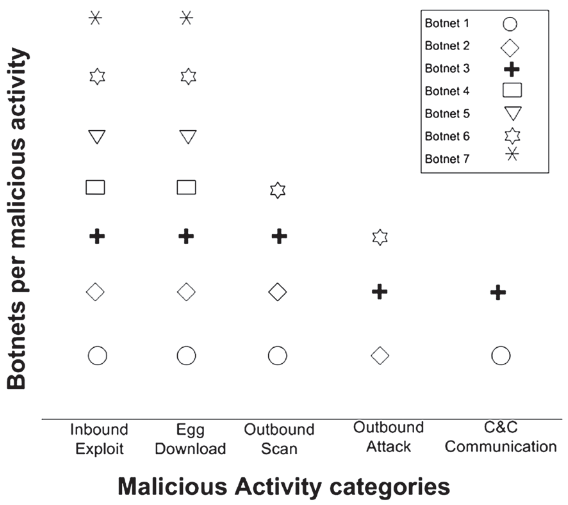
The datasets have varying numbers of botnets and widely varying numbers of bots in each botnet. They also have considerable overlap in the activity space. Figure 12 provides a feel for the datasets. Figure 12a shows the activities that botnets in the Dataset 3 are involved in. The strong overlap in activity space indicates that there are no distinct activities that can be used to segregate one botnet from another. Figure 12b shows the overlap in one activity, Egg Download, in the same dataset. The overlap in activity observed in Figure 12b is seen in the other activities in Dataset 3 too. Datasets 1 and 2 show similar patterns.

Table X shows the final results obtained using our approach on the bot infection profiles generated by BotHunter. The *Ncut* threshold was set to 0.98, which is higher than the thresholds used for Experiments 1 and 2. This is because BotHunter generates more succinct infection profiles, and we expect a higher degree of activity co-occurrences at the bots that belong to the same botnet. The results indicate that our approach can segment out the botnets with a high degree of accuracy. All the errors in mis-classification are from malicious hosts that have exhibited very low levels of activity, which makes it difficult to establish temporal co-occurrences with the activities at the other hosts.

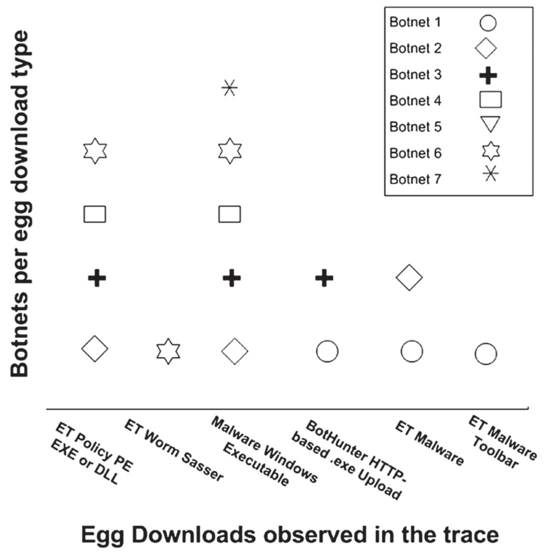
7. LIMITATIONS

As with all network security solutions, the proposed framework has some limitations, which are discussed here.

Our framework is based on the assumption that a botnet infection has resulted in the hosts exhibiting temporally co-occurring malicious activities. This is based on observations of botnet activities and their communication patterns in real networks [3,6]. If the bots are designed to engage in their pre-programmed behaviors in a random manner, our graph-based method as presented here will not work. But when the computers in a botnet do not act in any concerted and coordinated behavior, can the infected hosts still be considered to belong to a botnet? It is perhaps safe to assume that the bots behaving randomly and chaotically to avoid detection will also be of significantly diminished utility to a botmaster. Additionally, if there is no temporal co-occurrence in the malicious activities between the



(a)



(b)

Figure 12. (a) Activities exhibited by botnets in Dataset 3, and (b) overlap in 'Egg Download' in the same dataset.

Table X. Results on BotHunter traces.

Dataset	Botnets in trace	Botnets identified	No. of malicious hosts	No. mis-classified	Accuracy (%)
1	5	5	39	1	97.4
2	5	5	61	6	90.2
3	7	7	51	4	92.2

different hosts, it will be impossible for the infected hosts to mount, say, a DDoS attack on a target.

Another limitation of the algorithm as currently formulated is that we allow an infected host to engage in only one malicious activity at any given time. With the increase in computing power/cores in the machines of today, this assumption may no longer hold. However, it would be relatively straightforward to extend our approach to the case that allows for hosts to exhibit multiple malicious activities in any given time slot. To explain, assume that for a given edge, the node at one end exhibits a single activity in a given time slot but the node at the other end exhibits m activities simultaneously in the same time slot. When we get to updating the weight of this edge, instead of using just a single update of the sort described in our paper, we will now perform m such updates, one for each of the multiple activities seen. This can be further extended to the case when both ends exhibit multiple activities.

8. CONCLUSIONS

In this paper, we presented a graph-based framework for isolating botnets in a network. Our framework uses temporal co-occurrences in the activity space to detect botnets. This makes the framework independent of the software architecture of the malware infecting the hosts. Another advantage of our approach is that the newly infected hosts are automatically incorporated in the graph representation of the infected hosts that is maintained by our framework. By the same token, disinfected hosts are automatically dropped from the representation.

The proposed framework was validated by applying it to a simulated botnet, with the simulation based on a new model of botnets that unifies the current best understanding of the growth and decay of botnets, our best knowledge of the malicious activities they engage in, the various time constants associated with their activities, and so on. We also showed results on real traces generated by using real bot detection tools in infected networks.

On the strength of our validation, we believe that it is reasonable to conclude that one can use a graph-based framework, such as the one described here, to build an effective network-monitoring tool for the purpose of detecting and isolating botnets in a network.

REFERENCES

1. Gu G, Perdisci R, Zhang J, Lee W. Botminer: clustering analysis of network traffic for protocol- and structure-independent botnet detection. Proceedings of USENIX Security Symposium '2008, 2008; 139–154.
2. http://support.microsoft.com/botnets
3. Gu G, Zhang J, Lee W. BotSniffer: detecting botnet command and control channels in network traffic. Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08), 2008.

4. Zhao Y, Xie Y, Yu F, *et al.* BotGraph: large scale spamming botnet detection. Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, USENIX Association, 2009; 321–334.
5. Stringhini G, Holz T, Stone-Gross B, Kruegel C, Vigna G. BotMagnifier: locating spambots on the Internet. USENIX Security Symposium, 2011.
6. Rajab MA, Zarfoss J, Monrose F, Terzis A. A multifaceted approach to understanding the botnet phenomenon. IMC '06: Proceedings of the 6th ACM SIGCOMM on Internet Measurement, 2006; 41–52.
7. Ruitenbeek EV, Sanders WH. Modeling peer-to-peer botnets. Proceedings of the 2008 Fifth International Conference on Quantitative Evaluation of Systems, 2008; 307–316.
8. Dagon D, Gu G, Lee CP, Lee W. A taxonomy of botnet structures. In Proc. of the 23 Annual Computer Security Applications Conference (ACSAC'07), 2007.
9. Kim J, Radhakrishnan S, Dhall S. Measurement and analysis of worm propagation on internet network topology. Computer Communications and Networks, 2004. ICCCN 2004. Proceedings. 13th International Conference on, 2004; 495–500.
10. Stone-Gross B, Cova M, Cavallaro L, *et al.* Your botnet is my botnet: analysis of a botnet takeover. ACM Conference on Computer and Communications Security (CCS), 2009.
11. Holz T, Steiner M, Dahl F, Biersack E, Freiling F. Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. LEET'08: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, 2008; 1–9.
12. Freiling FC, Holz T, Wicherski G. Botnet tracking: exploring a root-cause methodology to prevent distributed denial-of-service attacks. ESORICS, 2005; 319–335.
13. Bailey M, Cooke E, Jahanian F, Xu Y, Karir M. A survey of botnet technology and defenses. *Conference for Homeland Security, Cybersecurity Applications and Technology* 2009; **0**: 299–304.
14. Wang Y, Chakrabarti D, Wang C, Faloutsos C. Epidemic spreading in real networks: an eigenvalue viewpoint. In SRDS, 2003; 25–34.
15. Staniford S, Paxson V, Weaver N. How to 0wn the Internet in your spare time. Proc. 11th USENIX Security Symposium, San Francisco, CA, 2002.
16. Goebel J, Holz T. Rishi: identify bot contaminated host by IRC nickname evaluation. Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets (HotBots), 2007.
17. Mazzariello C. IRC traffic analysis for botnet detection. Information Assurance and Security, International Symposium on, vol. 0] 2008; 318–323.
18. Huang Z, Zeng X, Liu Y. Detecting and blocking P2P botnets through contact tracing chains. *International Journal of Internet Protocol Technology* April 2010; **5**: 44–54.
19. Nagaraja S, Mittal P, Hong Cy, Caesar M, Borisov N. BotGrep: finding P2P bots with structured graph analysis. USENIX Security Symposium, 2010.
20. Choi H, Lee H, Lee H, Kim H. Botnet detection by monitoring group activities in DNS traffic. CIT '07: Proceedings of the 7th IEEE International Conference on Computer and Information Technology, 2007; 715–720.
21. Gu G, Porras P, Yegneswaran V, Fong M, Lee W. Bothunter: detecting malware infection through IDS-driven dialog correlation. SS'07: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, 2007; 1–16.
22. Bailey M, Oberheide J, Andersen J, Mao ZM, Jahanian F, Nazario J. Automated classification and analysis of internet malware. Proceedings of the 10th International Conference on Recent Advances in Intrusion Detection, RAID'07, 2007; 178–197.
23. Lu W, Tavallaee M, Ghorbani AA. Automatic discovery of botnet communities on large-scale communication networks. ASIACCS '09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, 2009; 1–10.
24. Dagon D, Zou CC, Lee W. Modeling botnet propagation using time zones. NDSS, 2006.
25. Hao S, Syed NA, Feamster N, Gray AG, Krasser S. Detecting spammers with snare: spatio-temporal network-level automatic reputation engine. Proceedings of the 18th Conference on USENIX Security Symposium, SSYM'09, 2009; 101–118.
26. http://mtc.sri.com/conficker/addendumc/
27. Shi J, Malik J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2000; **22**(8):888–905.