This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2532804, IEEE Transactions on Control of Network Systems

Botnet Detection based on Anomaly and Community Detection *

Jing Wang† and Ioannis Ch. Paschalidis‡

***Abstract*—We introduce a novel two-stage approach for the important cyber-security problem of detecting the presence of a botnet and identifying the compromised nodes (the bots), ideally before the botnet becomes active. The first stage detects anomalies by leveraging large deviations of an empirical distribution. We propose two approaches to create the empirical distribution: a flow-based approach estimating the histogram of quantized flows, and a graph-based approach estimating the degree distribution of node interaction graphs, encompassing both Erdős-Rényi graphs and scale-free graphs. The second stage detects the bots using ideas from social network community detection in a graph that captures correlations of interactions among nodes over time. Community detection is done by maximizing a modularity measure in this graph. The modularity maximization problem is non-convex. We propose a convex relaxation, an effective randomization algorithm, and establish sharp bounds on the suboptimality gap. We apply our method to real-world botnet traffic and compare its performance with other methods.**

***Index Terms*—Anomaly detection, cyber-security, botnets, social networks, random graphs, optimization.**

I. Introduction

A botnet is a network of compromised computers controlled by a "botmaster." Botnets are typically used for Distributed Denial-of-Service (DDoS) attacks, click fraud, or spamming. DDoS attacks flood the victim with packets/requests from many bots, effectively consuming critical resources and denying service to legitimate users. Botnet attacks are widespread. In a recent survey, 300 out of 1000 surveyed businesses have suffered from DDoS attacks and 65% of the attacks cause up to $10,000 loss per hour [1]. Both click fraud and spamming are harmful to the web economy.

Because of these losses, botnet detection has received considerable attention. Common intrusion detection focuses on individual hosts but is often ineffective in preventing botnet formation because not all hosts are zealously monitored and protected.

Early botnets used IRC, a protocol initially designed for Internet chat, to *Command and Control (C&C)* their bots (infected machines). As a result, a lot of botnet detection methods exploited this feature [2], [3]. Recently, though, botnets have evolved to bypass these detection methods by using more flexible C&C channels, such as `HTTP` and `P2P` protocols. In addition, more types of C&C channels are emerging, including `Twitter`.

Some methods have been proposed to handle these novel botnets with more flexible C&C mechanisms by analyzing the communication patterns among hosts. [4] proposes a method, named BotMagnifier, that deduces bots through their communication with a set of seed IPs. However, only spam bots can be handled by BotMagnifier and the seed IPs need to be given as input data. An alternative approach called BotHunter [5] models the infection process using a state transition diagram. A variety of methods are used to detect these transitions and determine whether a node is infected or not. Despite its popularity, BotHunter has the drawback that it cannot detect bots that were infected before the deployment of the system, and its infection state diagram can only describe a small set of bot behaviors.

In this paper, we propose a two-stage approach for botnet detection. The first stage detects and collects network anomalies that are associated with the presence of a botnet while the second stage identifies the bots by analyzing these anomalies (see Fig. 1). Our approach exploits the following two observations: (1) botmasters or attack targets are easier to detect because they communicate with many other nodes, and (2) the activities of infected machines are more correlated with each other than those of normal machines [3].

For the first stage, we propose two anomaly detection methods, both of which leverage the theory of large deviations [6]. Based on the stochastic model-free method proposed in [7], [8], [9], the first anomaly detection method quantizes flow-level data (e.g., Cisco NetFlows) and monitors the histogram of quantized flows. The second anomaly detection method aggregates packet-level data to graphs and monitors their empirical degree distribution. Compared to our preliminary work in [10] that considered only graph-based anomaly detection for Erdős-Rényi graphs, here we also handle scale-free graphs and draw on hypothesis testing to select the appropriate model.

For the second stage, we first identify a set of highly interactive nodes, which are referred to as *pivotal nodes*. Both botmasters and targets are very likely to

Research partially supported by the NSF under grants CNS-1239021, IIS-1237022, and CCF-1527292, by the ARO under grants W911NF-11-1-0227 and W911NF-12-1-0390, and by the ONR under grant N00014-10-1-0952.

† Center for Information and Systems Eng., Boston University, 15 St. Mary's St., Brookline, MA 02446, wangjing@bu.edu.

‡ Dept. of Electrical & Computer Eng., and Division of Systems Eng., Boston University, 8 Mary's St., Boston, MA 02215, yannisp@bu.edu, http://sites.bu.edu/paschalidis/.

2325-5870 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2532804, IEEE Transactions on Control of Network Systems

be *pivotal nodes* because they need to interact with bots frequently. These interactions correspond to C&C traffic for botmasters and attacking traffic for targets. In either case, the interactions between each bot and *pivotal nodes* are correlated. To characterize this correlation, we construct a *Social Correlation Graph (SCG)*, whose formal definition is in Sec. IV-B1. We can detect bots by detecting the community that exhibits high interaction with *pivotal nodes* in the SCG. We propose a novel community detection method based on a refined modularity measure. This problem is posed as a maximization of the modularity measure, which is NP-complete. We develop a convex relaxation scheme and establish bounds on its performance using ideas for the MAXCUT [11] problem.

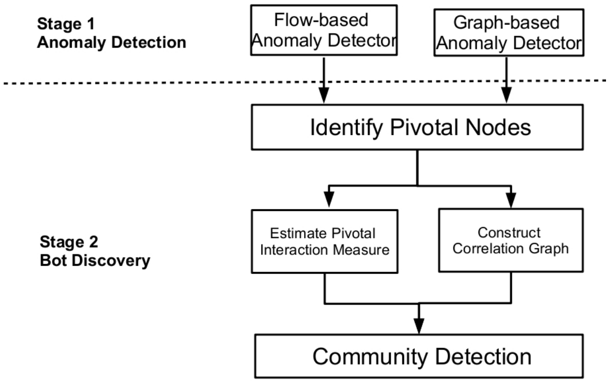

Figure 1. Overview of our method.

Notation: Throughout the paper all vectors are assumed to be column vectors. We use lower case boldface letters to denote vectors and for economy of space we write $\mathbf{x} = (x_0, \ldots, x_n)$ for the column vector $\mathbf{x}$. We define $[\mathbf{x}]_i = \sum_{j=0}^{i} x_j$. We use upper case boldface letters to denote matrices, script letters for sets, and denote by $|\mathcal{A}|$ the cardinality of set $\mathcal{A}$. log denotes the natural logarithm. $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix and $\succeq 0$ positive semi-definiteness. Throughout the paper we use n to denote the number of nodes in the network.

II. Large deviations of network processes

The anomaly detection stage (Stage 1) is based on analyzing network processes such as network flows and the degree of graphs representing node interactions. This section presents the large deviations results on which anomaly detection will be based. We start with a formal definition of the *Large Deviations Principle (LDP)* for a family of probability measures $\{\boldsymbol{\mu}^{(n)}\}$.

Definition 1
For every closed set $\mathcal{B}$ of probability vectors,

$$\lim_{n\to\infty} \sup \frac{1}{n} \log P_n(\boldsymbol{\mu}^{(n)} \in \mathcal{B}) \quad \leq \quad -\inf_{\boldsymbol{\mu}\in\mathcal{B}} I(\boldsymbol{\mu}),$$

and for every open set $\mathcal{B}$ of probability vectors, —

$$\lim_{n\to\infty} \inf \frac{1}{n} \log P_n(\boldsymbol{\mu}^{(n)} \in \mathcal{B}) \quad \geq \quad -\inf_{\boldsymbol{\mu}\in\mathcal{B}^\circ} I(\boldsymbol{\mu}),$$

where $\mathcal{B}^\circ$ denotes the interior of $\mathcal{B}$ and P_n is the probability law.

More intuitively, Def. 1 states that when n is large enough, the distribution P_n behaves as

$$P_n(\boldsymbol{\mu}^{(n)} \approx \boldsymbol{\mu}) \asymp e^{-nI(\boldsymbol{\mu})}. \tag{1}$$

The function $I(\boldsymbol{\mu})$ characterizes the *exponential decay rate* of this probability and is called the *rate* function.

A. LDP for discrete random variables

Given a discrete random variable X whose alphabet is $\Sigma = (\sigma_1, \ldots, \sigma_{|\Sigma|})$, the probability distribution of X can be written as a vector $\mathbf{p} = (p_1, \ldots, p_{|\Sigma|})$, where p_i is the probability of X being equal to σ_i.

Given n samples $\mathcal{X} = \{x_1, \ldots, x_n\}$ of X, the empirical distribution is a vector $\boldsymbol{\mu}^{(n)} = (\mu_1^{(n)}, \ldots, \mu_{|\Sigma|}^{(n)})$ where $\mu_i^{(n)} = (1/n)\sum_{j=1}^{n} 1(x_j = \sigma_i)$. $\boldsymbol{\mu}^{(n)}$ satisfies an LDP with rate function

$$I(\boldsymbol{\mu}) = D(\boldsymbol{\mu} \parallel \mathbf{p}), \tag{2}$$

where $D(\boldsymbol{\mu} \parallel \mathbf{p}) = \sum_i \mu_i \log(\mu_i/p_i)$ is the Kullback-Leibler (KL) divergence of two probability vectors [6].

B. LDP for the degree distribution of random graphs

Let $\mathbb{G}_n$ denote the space of all undirected graphs with n vertices. For any graph $\mathcal{G} \in \mathbb{G}_n$, let $\mathbf{d} = (d_1, \ldots, d_n)$ denote the labeled degree sequence of $\mathcal{G}$, with d_i denoting the degree of node i. Let $m = \frac{1}{2}\sum_{j=1}^{n} d_j$ denote the number of edges in $\mathcal{G}$. We assume that any two nodes are connected by at most one edge, which implies that the node degree in $\mathcal{G}$ is less than n.

For $0 \leq i \leq n-1$, let $h_i = \sum_{j=1}^{n} 1\,(d_j = i)$ be the number of vertices in $\mathcal{G}$ of degree i, where $1(\cdot)$ is the indicator function. Henceforth, $\mathbf{h} = (h_0, \ldots, h_{n-1})$, a quantity not dependent on the ordering of vertices, will be referred to as the *degree frequency vector* of graph $\mathcal{G}$. The empirical distribution of the degree sequence $\mathbf{d}$, defined by $\boldsymbol{\mu}^{(n)}$, is a probability measure on $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ that puts mass h_i/n at i, for $0 \leq i \leq n-1$. In the following sections, we present an LDP for empirical degree distributions $\boldsymbol{\mu}^{(n)}$ of two types of random graphs.

C. Erdős-Rényi model

In the Erdős-Rényi (ER) model, a graph is constructed by connecting nodes randomly. Each edge is included in the graph with probability p, independent from every other edge. We will use $G(n,p)$ to denote this model.

2325-5870 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

The distribution of the degree of any particular vertex v is binomial. Namely,

$$P(d_v = k) = \binom{n-1}{k} p^k (1-p)^{n-1-k}.$$

It it well known that when the number of nodes $n \to \infty$ and np is constant, the binomial distribution converges to the Poisson distribution. Let $\lambda = np$ denote the constant. Then, in the limiting case, the probability that the degree of a node is k equals

$$P_{ER}(k;\lambda) = p_{\lambda k} = \lambda^k e^{-\lambda}/k!, \tag{3}$$

which is independent of the node label. Let $\mathbf{p}_\lambda = (p_{\lambda 0}, \dots, p_{\lambda\infty})$ be the Poisson distribution viewed as a vector parametrized by λ.

Let $\mathbb{P}(\mathbb{N}_0)$ be the space of all probability measures defined on $\mathbb{N}_0$. We view any probability measure $\boldsymbol{\mu} \in \mathbb{P}(\mathbb{N}_0)$ as an infinite vector $\boldsymbol{\mu} = (\mu_0, \dots, \mu_\infty)$. Let $\mathcal{S} = \{\boldsymbol{\mu} \in \mathbb{P}(\mathbb{N}_0) \mid \bar{\mu} := \sum_{i=0}^{\infty} i\mu_i < \infty\}$ be the set of all probability measures on $\mathbb{N}_0$ with finite mean.

It is easy to verify that $\mathbf{p}_\lambda \in \mathcal{S}$. Let P_n denote the degree distribution of the ER model $G(n, \lambda/n)$ on the space $\mathbb{G}_n$. [12] proves that the ER model satisfies an LDP for the empirical degree distribution $\boldsymbol{\mu}^{(n)}$ in subsets of $\mathcal{S}$ under P_n with the following rate function.

Definition 2
For the ER model $G(n, \lambda/n)$ define the rate function $I_{ER} : \mathcal{S} \to [-\infty, \infty]$ as

$$I_{ER}(\boldsymbol{\mu}; \lambda) = \\ D(\boldsymbol{\mu} \parallel \mathbf{p}_\lambda) + \frac{1}{2}(\bar{\mu} - \lambda) + \frac{\bar{\mu}}{2}\log\lambda - \frac{\bar{\mu}}{2}\log\bar{\mu},$$

where $D(\boldsymbol{\mu} \parallel \mathbf{p}_\lambda) = \sum_i \mu_i \log(\mu_i/p_{\lambda i})$ is the KL divergence of $\boldsymbol{\mu}$ with respect to $\mathbf{p}_\lambda$.

D. Preferential attachment model

Preferential Attachment (PA) processes are graph networks that evolve in time by linking new nodes progressively to existing nodes, where the probability of each existing node to be linked depends on its degree [13]. We view a PA process as a sequence of random graphs $\mathscr{G} = \{\mathcal{G}_1, \dots, \mathcal{G}_n\}$, where $\mathcal{G}_j$ is the random graph at time j. We assume that only one new node is attached each time, i.e., $|\mathcal{G}_{j+1}| - |\mathcal{G}_j| = 1$ for all $j = 1, \dots, n-1$.

Initially, the graph $\mathcal{G}_1$ consists of two nodes with a single (undirected) edge between them. At time $j+1$, a new node is linked to node $x \in \mathcal{G}_j$ with probability proportional to some function $w(d_x(j))$, where $d_x(j)$ is the degree of node x at time j. The graph is called "scale-free" when $w(d) = d + \alpha$ and $\alpha > -1$. Here, "scale-free" means that the degree distribution of $\mathcal{G}_n$ converges to a power-law (i.e., $\mu_k^{(n)} \sim k^{-\theta}$ for some $\theta > 0$) as $n \to \infty$.

1) Generalized Polya urn model: The evolution of the degree distribution in the PA model is equivalent to a *generalized Polya urn model* [14] $\mathscr{U} = \{\mathcal{U}_0, \dots, \mathcal{U}_n\}$ as follows. Initially, $\mathcal{U}_0$ has two empty urns. Suppose $p(t) : [0,1] \to [0,1]$ and $\beta(t) : [0,1] \to [0,\infty)$ are two given functions. At each time $j+1$, we first add a new urn with no ball to the collection and then

1) with probability $p(j/n)$, we place a new ball in this new urn;
2) with probability $1 - p(j/n)$, we place a new ball in one of the other urns $x \in \mathcal{U}_j$, selecting x with probability proportional to $w(b_x) = b_x + \beta(j/n)$, where b_x is the number of balls in the urn x.

To make the connection between this model and a graph, associate an urn with k balls with a node of the graph with degree $k+1$. Initially, $\mathcal{U}_0$ corresponds to a graph with two connected nodes, each with degree 1. At each time we add a new node and connect it to an existing node. Placing a ball in an urn with no balls, is equivalent to connecting the new node to an existing node of degree 1, ending up with one node of degree 2 and one node with degree 1. Placing a ball in an existing urn x with b_x balls is equivalent to connecting a node of degree 1 (an urn with no balls like the one introduced each time) with an existing node whose degree increases by 1.

Suppose $\boldsymbol{\mu} = (\mu_0, \dots, \mu_n)$ is the degree distribution we observe at last (as $t = 1$). We consider two special cases of the PA model whose degree distributions satisfy a power-law asymptotically [13].

2) Offset Barabási-Albert (BA) process: If $p(t) \equiv 0$, $\beta(t) \equiv 1 + \alpha$, then the generalized urn model becomes a so-called "offset" BA process with selection function $w(d) = d + \alpha$. An "offset" BA process generates trees with no cycles. In this model, the probability for a node to have degree k is $P_{BA}(k;\alpha)$. It has been shown that asymptotically ($n \to \infty$) [13]

$$P_{BA}(k;\alpha) \asymp k^{-(\alpha+3)}/\zeta(\alpha+3),$$

where $\zeta(x) = \sum_{k=1}^{\infty} k^{-x}$ is Riemann's zeta-function.

We can obtain that the empirical degree distribution of the offset BA process satisfies a sample-path LDP with rate function

$$I_{BA}(\boldsymbol{\mu};\alpha) = \sum_{i\geq 0}(1 - [\boldsymbol{\mu}]_i) \log \frac{1 - [\boldsymbol{\mu}]_i}{(i+1+\alpha)\mu_i/(2+\alpha)} \\ + (1 - \textstyle\sum_{i\geq 0} i\mu_i)\log(2+\alpha), \tag{4}$$

where $[\boldsymbol{\mu}]_i = \sum_{j=0}^{i} \mu_i$ (see Appendix A).

3) Chung-Handjani-Jungreis (CHJ) model: If $p(t) \equiv p$ and $\beta(t) \equiv 1$, the generalized urn model corresponds to an attachment scheme that can generate graphs instead of trees. This model is a special case of the CHJ model [15] with asymptotic degree distribution

$$P_{CHJ}(k;p) \asymp k^{-\left(1+(1-p)^{-1}\right)}/\zeta(1+(1-p)^{-1}).$$

2325-5870 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

We can obtain (see Appendix A) that the empirical degree distribution satisfies a sample-path LDP with rate function

$$\begin{aligned} I_{CHJ}(\boldsymbol{\mu}; p) &= (1-\mu_0)\log\frac{1-\mu_0}{p+(1-p)\mu_0/2} \\ &+ \textstyle\sum_{i=1}^{\infty}(1-[\boldsymbol{\mu}]_i)\log\frac{1-[\boldsymbol{\mu}]_i}{(1-p)(i+1)\mu_i/2} \\ &+ (1-\textstyle\sum_{i\geq 0} i\mu_i)\log\frac{2}{1-p}. \end{aligned} \quad (5)$$

III. Anomaly detection

In this section, we propose two approaches for anomaly detections, both based on the results presented in the previous section. The first approach, which takes netflow files as input, quantizes flows and treats the quantized flows as observations of a discrete random variable. The second approach, which takes pcap files as input, aggregates packets to graphs and leverages the LDP for the graph degree distribution. A *Generalized Likelihood Ratio Test (GLRT)* is proposed to select the most appropriate random graph model.

A. Flow-based approach

NetFlow offers a concise representation of network traffic and has become a de-facto industry standard. Each flow describes a communication session, whose duration varies from seconds to days.

The overall idea of our approach is to quantize flows and to treat them as independent observations of a discrete random variable. We first cluster network addresses (part of the flow characteristics) into a manageable number of clusters. For simplicity of notation, we only consider IPv4 addresses. If $\mathbf{x}^i = (x_1^i, x_2^i, x_3^i, x_4^i) \in \{0, \ldots, 255\}^4$ and $\mathbf{x}^j = (x_1^j, x_2^j, x_3^j, x_4^j) \in \{0, \ldots, 255\}^4$ are two IPv4 addresses, a distance metric between them is defined as $d(\mathbf{x}^i, \mathbf{x}^j) = \sum_{k=1}^{4} 256^{(4-k)}|x_k^i - x_k^j|$. Letting $\mathcal{X}$ be the set of unique IP addresses, we apply typical K-means clustering on $\mathcal{X}$ using the distance metric mentioned above. A flow will be represented by the following features: (1) the cluster label of the source IP address, (2) the cluster label of the destination IP address, (3) the source port number, (4) the destination port number, (5) the flow duration, (6) the data bytes sent from source to destination, and (7) the data bytes sent from destination to source.

Suppose the input is a sequence of flows $\mathcal{F} = \{\mathbf{f}^1, \ldots, \mathbf{f}^n\}$. For each flow $\mathbf{f}$, we quantize each feature separately and denote by $\sigma(\mathbf{f})$ the "type" of the quantized flow while Σ is the corresponding alphabet. For any $\rho \in \Sigma$, the empirical measure is

$$\mu_{\mathcal{F}}(\rho) = (1/n)\textstyle\sum_{i=1}^{n} 1(\sigma(\mathbf{f}^i) = \rho). \quad (6)$$

We write $\boldsymbol{\mu}_{\mathcal{F}} = \{\mu_{\mathcal{F}}(\rho) : \rho \in \Sigma\}$ for the vector empirical measure.

Let $\boldsymbol{\mu}_{ref}$ denote the probability vector calculated from some reference flows $\mathcal{F}_{ref}$ and let $I_{flow}(\boldsymbol{\mu}) = D(\boldsymbol{\mu} \parallel \boldsymbol{\mu}_{ref})$. Using the LDP presented in Section II-A, we propose the following anomaly detector:

$$\mathcal{I}_{flow}(\mathcal{F}) = 1(I_{flow}(\boldsymbol{\mu}_{\mathcal{F}}) \geq \xi), \quad (7)$$

where ξ is a detection threshold. It was shown in [7] that (7) is asymptotically Neyman-Pearson optimal. We group flows into windows based on their timestamps and apply the anomaly detector above for each window.

B. Graph-based approach

We also propose a graph-based approach that processes packet-level data. We treat each packet as an interaction record between the source and the destination. We first group packets into windows based on their timestamps. For all k, we denote by $\mathcal{W}_k$ the collection of packets in window k. We define the *interaction graph* for window k as follows.

Definition 3

(Interaction Graph). *Let $\mathcal{E}_k$ be an edge set such that $(i, j) \in \mathcal{E}_k$ if there exists at least one packet $\mathtt{r} \in \mathcal{W}_k$ whose source is node i and destination is node j or source is node j and destination is node i. Then the interaction graph $\mathcal{G}_k = (\mathcal{V}, \mathcal{E}_k)$ corresponding to $\mathcal{W}_k$ is an undirected graph with vertex set $\mathcal{V}$, the set of all nodes in the network, and edge set $\mathcal{E}_k$.*

1) Model selection: Since the LDP rate function in random graphs depends on the graph model that is used, we first present a method to select the appropriate graph model. We assume M independent observations of node degrees $\mathscr{D} = \{d_1, \ldots, d_M\}$ under normal operation of the network. In practice, we could pick l nodes in the network randomly as monitoring points and observe the degrees of these nodes in K different *interaction graphs*. This will lead to $M = Kl$ samples. Note that the observations at those l nodes may not be independent. Yet, the dependence is negligible if $l \ll n$.

For the ER model with degree distribution $P_{ER}(k; \lambda)$ (cf. (3)) the log-likelihood of $\mathscr{D}$ is:

$$L_{ER}(\mathscr{D}; \lambda) = (\log\lambda)\textstyle\sum_{i=1}^{M} d_i - \lambda M + C, \quad (8)$$

where $C = -\sum_i \log(d_i!)$.

For both the offset BA model and the CHJ model, the asymptotic degree distribution is a power-law with the form $P_{PA}(k; \gamma) = k^{-\gamma}/\zeta(\gamma)$, where γ is a parameter. The log-likelihood of $\mathscr{D}$ is:

$$L_{PA}(\mathscr{D}; \gamma) = -\gamma\textstyle\sum_{i=1}^{M}\log d_i - M\log\zeta(\gamma). \quad (9)$$

2325-5870 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2532804, IEEE Transactions on Control of Network Systems

In practice, we may observe some isolated nodes i whose degree $d_i = 0$. In this case, $\log d_i = -\infty$ and the PA model is completely ruled out. However, these isolated nodes may be the result of observation error. Instead of ruling out the PA model completely, we add a finite penalty θ for each isolated node. Namely, we write

$$L_{PA}(\mathscr{D};\gamma) = L^{+}_{PA}(\mathscr{D};\gamma) - \theta {\textstyle\sum_i} 1(d_i = 0), \qquad (10)$$

where L^{+}_{PA} is simply the same as in (9) with the summation taken only over i with $d_i > 0$. An appropriate value of θ can be determined through experiments, as we will elaborate in Sec. V-D.

We use the *Generalized Likelihood Ratio Test (GLRT)* (see [16] for related discussion) to select the reference model. Let $\mathcal{H}_0$ be the hypothesis that the ER model is the most appropriate and $\mathcal{H}_1$ be the alternative hypothesis; then the GLRT is

$$\max_{\lambda} L_{ER}(\mathscr{D};\lambda) - \max_{\gamma} L_{PA}(\mathscr{D};\gamma) \underset{\mathcal{H}_1}{\overset{\mathcal{H}_0}{\gtrless}} \eta, \qquad (11)$$

where η is a prescribed threshold.

The Maximum Likelihood Estimate (MLE) for the parameters of the ER and PA models can be calculated easily by maximizing (8) and (9), respectively. Setting the derivative of $L_{ER}(\mathscr{D};\lambda)$ to zero, the MLE for the parameter of the ER model is $\hat{\lambda} = (\sum_{i=1}^{M} d_i)/M$. Similarly, letting $\phi(x) = \dot{\zeta}(x)/\zeta(x)$, where dot denotes derivative, and $\phi^{-1}(x)$ be the inverse function of $\phi(x)$, the MLE for the parameter of the PA model is $\hat{\gamma} = \phi^{-1}(-(\sum_{d_i>0} \log d_i)/M)$.

If the ER model is selected, we use $I_{ER}(\boldsymbol{\mu};\hat{\lambda})$ as the rate function. Let now $\hat{\alpha}$ be the MLE of the offset BA process and $\hat{p}$ be the MLE of the CHJ model; then $\hat{\alpha} = \hat{\gamma} - 3$ and $\hat{p} = 1 - (1-\hat{\gamma})^{-1}$, respectively. We can use the following combined rate function

$$I_{PA}(\boldsymbol{\mu};\hat{\gamma}) = \min(I_{BA}(\boldsymbol{\mu};\hat{\alpha}), I_{CHJ}(\boldsymbol{\mu};\hat{p})) \qquad (12)$$

if the PA model is selected. It can be shown that $I_{PA}(\boldsymbol{\mu};\hat{\gamma})$ is the rate function for a random graph model that is either the offset BA model or the CHJ model (see Appendix B).

2) A formal anomaly detection test: Next, we consider the problem of evaluating whether a graph $\mathcal{G}$ is normal, i.e., comes from either the ER model or the PA model with desirable parameters ($\mathcal{H}_0$). Let $\boldsymbol{\mu}_{\mathcal{G}}$ be the empirical degree distribution of the graph $\mathcal{G}$. $I(\boldsymbol{\mu}_{\mathcal{G}})$ in Definition 1 provides a rigorous indicator of the normality of $\boldsymbol{\mu}_{\mathcal{G}}$. The rate function that should be used depends on the result of the model selection procedure. If the ER model is selected, we use $I_{ER}(\boldsymbol{\mu};\hat{\lambda})$ as the rate function; otherwise, we use $I_{PA}(\boldsymbol{\mu};\hat{\gamma})$.

Let

$$I(\boldsymbol{\mu}) = \begin{cases} I_{ER}(\boldsymbol{\mu},\hat{\lambda}), & \text{if ER is selected,} \\ I_{PA}(\boldsymbol{\mu},\hat{\gamma}), & \text{if PA is selected.} \end{cases}$$

The graph-based anomaly detector is:

$$\mathcal{I}_{graph}(\mathcal{G}) = 1(I(\boldsymbol{\mu}_{\mathcal{G}} \geq \xi), \qquad (13)$$

where ξ is a detection threshold. The test in (13) is a generalized *Hoeffding test*. Since $I(\boldsymbol{\mu}_{\mathcal{G}})$ satisfies an LDP regardless of the selected model, the generalized Hoeffding test (13) satisfies the Generalized Neyman-Pearson (GNP) criterion [6].

IV. Botnet discovery

The network anomaly detection technique in the previous section can only report whether there is anomaly or not. In this section, we present a *botnet discovery* technique that can identify botnets. We apply a sliding window to network traffic, monitor windows continuously and store all anomalies. The botnet discovery technique can be applied when the number of detected anomalies is large enough.

The flow-based anomaly detector reports abnormal windows and the graph-based anomaly detector reports abnormal interaction graphs. We first unify the output of the two methods by defining *interaction records*.

An *interaction record* (t_s, i, j) is a tuple of timestamp, source IP and destination IP, and it represents that node i and j interact at time t_s (regardless of direction). For the flow-based detector, an anomalous window can be represented as a set of interaction records because each flow can be easily converted to an interaction record. For the graph-based detector, an interaction graph can be also represented as a set of interaction records because each edge corresponds to an interaction record. As a result, in both methods, an anomaly can be represented as a set of interaction records $\mathcal{S} = \{r_1, \ldots, r_{|\mathcal{S}|}\}$. The input of our botnet discovery method is a sequence of anomalies $\mathcal{A} = \{\mathcal{S}_1, \ldots, \mathcal{S}_{|\mathcal{A}|}\}$.

A. Identification of pivotal nodes

Detecting bots directly is non-trivial. Instead, detecting the leaders (botmasters) or targets is simpler because they are more interactive than normal nodes. Botmasters need to "command and control" their bots to maintain the botnet, and bots actively interact with victims in a typical DDoS attack, even before the attack while they probe the targets. Both leaders and targets, henceforth referred to as *pivotal nodes*, are highly interactive. Suppose the total number of nodes that appear in $\mathcal{A}$ is n. Let G^{ij}_k be the number of interactions between node i and node j in anomaly $\mathcal{S}_k$. Then,

$$e_i = (1/|\mathcal{A}|) {\textstyle\sum_{k=1}^{|\mathcal{A}|} \sum_{j=1}^{n}} G^{ij}_k, \qquad i = 1, \ldots, n, \quad (14)$$

represents the amount of interaction of node i with all other nodes in $\mathcal{A}$ and will be called the *total interaction measure* of node i. We next define *pivotal nodes*.

2325-5870 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2532804, IEEE Transactions on Control of Network Systems

Definition 4
*(*Pivotal nodes*). The set of* pivotal nodes *is* $\mathcal{N} = \{i : e_i > \tau\}$*, where* τ *is a threshold.*

After identifying *pivotal nodes*, we turn to detecting the community associated with *pivotal nodes*.

B. Botnet discovery

We now focus on interactions between pivotal nodes and the remaining nodes. Pivotal nodes are either botmasters or attack targets; in either case their interactions with the bots are likely to be correlated. This section presents a technique that detects a community of highly correlated and highly interactive nodes.

Compared to similar approaches in community detection, e.g., the leader-follower algorithm [17], our method takes advantage of not only temporal features (amount of interaction) but also correlation relationships. These relationships are characterized using a graph, whose definition is presented next.

1) Construction of the Social Correlation Graph: For $i = 1, \ldots, n$, let X_i represent the number of interactions between node i and pivotal nodes. For each anomaly $\mathcal{S}_k \in \mathcal{A}$, we obtain a sample of X_i as $x_i^k = \sum_{j \in \mathcal{N}} G_k^{ij}$. Let $\bar{X}_i = (1/|\mathcal{A}|) \sum_{k=1}^{|\mathcal{A}|} x_i^k$ be the sample mean and $\sigma(X_i) = \sqrt{(1/(|\mathcal{A}|-1)) \sum_{k=1}^{|\mathcal{A}|} (x_i^k - \bar{X}_i)^2}$ be the sample standard deviation of X_i for all i. The sample *correlation coefficient* is defined as

$$\rho(X_i, X_j) = \frac{\sum_{k=1}^{|\mathcal{A}|} [(x_i^k - \bar{X}_i)(x_j^k - \bar{X}_j)]}{(|\mathcal{A}|-1)\sigma(X_i)\sigma(X_j)},$$

with the convention that $\rho(X_i, X_j) = 0$ if $\sigma(X_i) = 0$ or $\sigma(X_j) = 0$. By definition, $\rho(X_i, X_j) \in [-1, 1]$.

Definition 5
(Social Correlation Graph*). The Social Correlation Graph (SCG)* $\mathcal{C} = (\mathcal{V}, \mathcal{E}_c)$ *is an undirected graph with vertex set* $\mathcal{V}$ *and edge set* $\mathcal{E}_c = \{(i, j) : |\rho(X_i, X_j)| > \tau_\rho\}$*, where* τ_ρ *is a prescribed threshold.*

Because the behaviors of bots are correlated, they are more likely to be connected to each other in the SCG. Our problem is to find an appropriate division of the SCG to separate bots and normal nodes. Our criterion for "appropriate" is related to the well-known concept of *modularity* in community detection [18], [19].

2) Modularity-based community detection: The problem of community detection in a graph amounts to dividing its nodes into non-overlapping groups such that connections within groups are relatively dense while those between groups are sparse [18]. The *modularity* for a given subgraph is defined to be the fraction of edges within the subgraph minus the expected fraction of such edges in a randomized null model. This measure has inspired a broad range of community detection methods named as *modularity-maximization* methods.

We consider the simpler case when there is only one botnet in the network. As a result, we want to divide the nodes into two groups, one for bots and one for normal nodes. Define s_i as:

$$s_i = \begin{cases} 1, & \text{node } i \text{ is a bot,} \\ -1, & \text{otherwise.} \end{cases}$$

Let d_i^c be the degree of node i in SCG $\mathcal{C} = (\mathcal{V}, \mathcal{E}_c)$ for $i = 1, \ldots, n$ and $m^c = (1/2) \sum_i d_i^c$ the number of edges in $\mathcal{C}$. For a partition specified by $\mathbf{s} = (s_1, \ldots, s_n)$, its *modularity* is defined as in [18]

$$Q(\mathbf{s}) = (1/2m^c) {\textstyle\sum}_{i,j=1}^{n} (A_{ij} - N_{ij}) \delta(s_i, s_j), \quad (15)$$

where $\delta(s_i, s_j) = (1/2)(s_i s_j + 1)$. It is easy to observe that $\delta(s_i, s_j)$ is an indicator of whether node i and node j are of the same type. $A_{ij} = 1(|\rho(X_i, X_j)| > \epsilon_\rho)$ is an indicator of the adjacency of nodes i and j. N_{ij} is the *expected number of edges* between node i and node j in a null model. The selection of the null model is empirical; a common choice [19] is to assume that an arbitrary edge attaches to node i with probability κd_i^c, where κ is a normalization constant. Then, $N_{ij} = \kappa^2 d_i^c d_j^c$. Setting $\sum_{i,j} N_{ij} = 2m^c$, we can solve for κ and obtain $N_{ij} = d_i^c d_j^c / (2m^c)$. The optimal division of vertices should maximize the modularity measure (15).

3) Refined modularity: We introduce two refinements to the *modularity* measure to make it suitable for botnet detection. First, intuitively, bots should have strong interactions with *pivotal nodes* and normal nodes should have weaker interactions. We want to maximize the difference. To capture the amount of interaction between node i and *pivotal nodes* define

$$r_i = (1/|\mathcal{A}|) {\textstyle\sum}_{k=1}^{|\mathcal{A}|} {\textstyle\sum}_{j \in \mathcal{N}} e_j G_k^{ij}. \quad (16)$$

We refer to r_i as *pivotal interaction measure* of node i. Then $\sum_i r_i s_i$ quantifies the difference between the *pivotal interaction measure* of bots and that of normal nodes. A natural extension of the modularity measure is to include an additional term to maximize $\sum_i r_i s_i$.

Second, the *modularity* measure is criticized to suffer from low resolution, i.e., it favors large communities and ignores small ones [20]. The botnet, however, could possibly be small. To address this issue, we introduce a regularization term for the size of botnets. Notice that $\sum_i 1(s_i = 1) = \sum_i (s_i + 1)/2$ is the number of detected bots. Thus, our refined *modularity* measure is

$$Q_d(\mathbf{s}) = \frac{1}{2m^c} {\textstyle\sum}_{i,j \in \mathcal{V}} \left(A_{ij} - \frac{d_i^c d_j^c}{2m^c} \right) s_i s_j + w_1 {\textstyle\sum}_i r_i s_i - w_2 {\textstyle\sum}_i \frac{s_i + 1}{2}, \quad (17)$$

2325-5870 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2532804, IEEE Transactions on Control of Network Systems

where w_1 and w_2 are appropriate weights.

The two modifications also influence the results of isolated nodes with degree 0, which possibly exist in SCGs. By Def. 5, node i becomes isolated if $\sigma(X_i) = 0$ or its correlations with other nodes are small enough. The placement of isolated nodes, however, does not influence the traditional modularity measure, resulting in arbitrary community detection results [18]. This limitation is addressed by the two additional terms. If node i is isolated and $r_i = 0$, then $s_i = -1$ in the solution because of the regularization term $w_2 \sum_i (s_i + 1)/2$. On the contrary, if r_i is large enough, $s_i = 1$ in the solution because of the term $w_1 \sum_i r_i s_i$.

4) Relaxation of the optimization problem: The modularity-maximization problem has been shown to be NP-complete [21]. The existing algorithms can be broadly categorized into two types: (i) heuristic methods that solve this problem directly [22], and (ii) mathematical programming methods that relax it into an easier problem first [21]. We follow the second route because it is more rigorous.

We define the modularity matrix $\mathbf{M} = \{M_{ij}\}_{i,j=1}^n$, where $M_{ij} = A_{ij}/(2m^c) - d_i^c d_j^c/(2m^c)^2$. Let $\mathbf{s} = (s_1, \ldots, s_n)$ and $\mathbf{r} = (r_1, \ldots, r_n)$; then the modularity-maximization problem becomes

$$\begin{aligned} f^* = \max \quad & \mathbf{s}'\mathbf{M}\mathbf{s} + \left(w_1\mathbf{r}' - \frac{w_2}{2}\mathbf{1}'\right)\mathbf{s} \\ \text{s.t.} \quad & s_i^2 = 1, \end{aligned} \tag{18}$$

where $\mathbf{1}$ is the vector of all ones and f^* denotes the optimal value. (18) is not necessarily convex. Letting

$$\mathbf{S} = \mathbf{s}\mathbf{s}', \qquad \mathbf{q}_0 = w_1\mathbf{r} - (w_2/2)\mathbf{1},$$

noticing that $\mathbf{s}'\mathbf{M}\mathbf{s} = \mathrm{Tr}(\mathbf{S}\mathbf{M})$, and by relaxing the constraint $\mathbf{S} = \mathbf{s}\mathbf{s}'$ into $\mathbf{S} - \mathbf{s}\mathbf{s}' \succeq 0$, we can write the following convex relaxation [23] of (18):

$$\begin{aligned} f^*_{\text{relax}} = \max \quad & \mathrm{Tr}(\mathbf{S}\mathbf{M}) + \mathbf{q}_0'\mathbf{s} \\ \text{s.t.} \quad & \begin{bmatrix} \mathbf{S} & \mathbf{s} \\ \mathbf{s}' & 1 \end{bmatrix} \succeq 0, \\ & S_{ii} = 1, \qquad \forall i. \end{aligned} \tag{19}$$

Let

$$\mathbf{X} = \begin{bmatrix} \mathbf{S} & \mathbf{s} \\ \mathbf{s}' & 1 \end{bmatrix}, \text{ and } \mathbf{W} = \begin{bmatrix} \mathbf{M} & \mathbf{q}_0/2 \\ \mathbf{q}_0'/2 & 0 \end{bmatrix};$$

then the problem can be compactly written as:

$$\begin{aligned} f^*_{\text{relax}} = \max \quad & \mathrm{Tr}\,(\mathbf{X}\mathbf{W}) \\ \text{s.t.} \quad & \mathbf{X} \succeq 0, \\ & X_{ii} = 1, \qquad \forall i. \end{aligned} \tag{20}$$

The problem above is a *Semi-Definite Programming* (SDP) problem and produces an upper bound on the optimal value of the original problem (18). It is well known that SDPs are polynomially solvable and many solvers are available.

Theorem IV.1 *Let χ^-_{min} be the smallest negative eigenvalue of $\mathbf{W}$, where $\chi^-_{min} = 0$ when $\mathbf{W} \succeq 0$. Then,*

$$\frac{2}{\pi} f^*_{relax} + \left(1 - \frac{2}{\pi}\right) \chi^-_{min}(n+1) \le f^* \le f^*_{relax}.$$

Proof: See Appendix C. ■

In the special case that $\mathbf{W} \succeq 0$, (18) becomes a MAXCUT problem and the lower bound degenerates to the well-known bound for MAXCUT [11].

5) Randomization: Solving the SDP relaxation (19), we obtain an optimal solution $(\mathbf{S}^*, \mathbf{s}^*)$ of (19) together with bounds on f^*. However, this solution may not be feasible for (18). To generate feasible solutions we use a randomization technique.

Notice that by feasibility in (19), $\mathbf{S}^* - \mathbf{s}^*(\mathbf{s}^*)' \succeq 0$ and it can be interpreted as a covariance matrix. If we pick $\mathbf{y}$ as a Gaussian random vector with $\mathbf{y} \sim N(\mathbf{s}^*, \mathbf{S}^* - \mathbf{s}^*(\mathbf{s}^*)')$, then $\mathbf{y}$ will solve the non-convex (18) "on average" over this distribution. Thus, to generate good feasible solutions of (18), we can sample $\mathbf{y}$ from $N(\mathbf{s}^*, \mathbf{S}^* - \mathbf{s}^*\mathbf{s}^{*'})$ and project to the feasible set as in $\hat{\mathbf{y}} = \mathrm{sgn}(\mathbf{y})$. Sampling a large number of points (e.g., 10,000) and keeping the one with the largest objective value can produce near-optimal solutions.

V. Experimental results

Our experimental results include two parts. In the first part, we compare our method with the BotHunter method [5] on the CTU-13 botnet dataset [24]. In the second part, we apply our method to a dataset generated by mixing a DDoS attack traffic dataset with real-world background traffic. In this part, we mainly focus on the performance of our *botnet discovery* approach.

A. Description of CTU-13 dataset

The CTU-13 is a dataset of botnet traffic that was captured in the Czech Technical University [24]. It contains 13 scenarios with various botnet types. We consider scenarios 1, 2, 6, 8, and 9 of the dataset.

- Scenario 1 corresponds to an IRC-based botnet that sent spam for almost 6.5 hours.
- Scenario 2 (2.5-hours) is from the same botnet.
- Scenario 6 is from a botnet that scanned SMTP (Simple Mail Transfer Protocol) servers for 2 hours and connected to several RDP (Remote Desktop Protocol) services. Different with Scenarios 1 and 2, this botnet neither sent spam nor did it attack. It's C&C server used a proprietary protocol.
- Scenario 8 is from a botnet that contacted a lot of Chinese C&C hosts and received large amounts

2325-5870 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2532804, IEEE Transactions on Control of Network Systems

of encrypted data. The botnet also cracked the passwords of machines during the 19-hour attack.

- Scenario 9 corresponds to a case where 10 local hosts were infected by a spamming botnet. More than 600 spams were sent over 5 hours.

For all the scenarios, the authors of the CTU-13 dataset convert the captured pcap files to NetFlows and release the processed flows. The dataset contains ground-truth labels for flows as follows: flows from or to the infected machines are labeled as "botnet"; flows from or to non-infected machines are labeled as "normal"; all other flows are labeled as "background."

B. Performance metrics for botnet detection

Typical performance metrics used in the literature are mostly flow-based (e.g., detection and false alarm probability for an anomalous flow) and do not usually account for the speed of detection. For botnets, we are interested in performance metrics that account for false alarms and miss-detections of bots but also recognize that early detection is useful.

In this paper, we use the performance metrics proposed by [24], which are defined based on host IPs classified either as bot or normal. Bot IPs are the IPs that either send or receive at least one "botnet" flow. We divide the data into multiple time frames, indexed by t, which are only used to compute performance metrics and are independent of the detection methods. For some metrics we will use a discount function (we use $\alpha = 0.01$)

$$c(t) = e^{-\alpha t} + 1, \tag{21}$$

to account for the fact that early (small t) detections or misses are more valuable than later ones. Let $N_b(t)$ be the number of bot IPs and $N_n(t)$ the number of normal IPs in time frame t, respectively. We define:

- $\overline{TP}(t) = TP(t)c(t)/N_b(t)$, where $TP(t)$ is the number of bot IPs correctly identified.
- $\overline{FN}(t) = FN(t)c(t)/N_b(t)$, where $FN(t)$ is the number of bot IPs missed.
- $\overline{FP}(t) = FP(t)/N_n(t)$, where $FP(t)$ is the number of normal IPs reported as bot IPs.
- $\overline{TN}(t) = TN(t)/N_n(t)$, where $TN(t)$ is the number of normal IPs reported as normal,

corresponding to true positives, false negatives, false positives, and true negatives, respectively.

Let now $tTP = \sum_t \overline{TP}(t)$, $tFN = \sum_t \overline{FN}(t)$, $tFP = \sum_t \overline{FP}(t)$, $tTN = \sum_t \overline{TN}(t)$ and define

- $FPR = \frac{tFP}{tTN+tFP}$,
- $Recall = \frac{tTP}{tTP+tFN}$,
- $Precision = \frac{tTP}{tTP+tFP}$,
- F1-Measure $= 2\frac{Precision*Recall}{Precision+Recall}$,
- G-Measure $= \sqrt{Precision * Recall}$.

The metrics above have similar semantic meanings with their classical counterparts [24].

C. Results on the CTU-13 dataset

The BotHunter method is based on a state-based infection sequence model and assumes the behavior of a bot machine can be described by a state diagram. It consists of two stages. The first stage identifies warnings based on anomaly and signature detection. In the second stage, the warnings are tracked over a temporal window, each contributing to an overall infection sequence score that is maintained for each host.

For the evaluation, we used the GAD software package we developed for evaluating anomaly detection methods [25], to which we added botnet evaluation functionalities. For the CTU-13 dataset we used our flow-based anomaly detection coupled with our botnet discovery approach as described in Sec. IV. The time frame size for evaluation is 5 minutes. For every scenario, the first 25 minutes of the dataset have no botnet traffic and are used for training. Each detection window is 2 seconds. The threshold for constructing our social correlation graph is $\tau_\rho = 10$. The regularization coefficients for interactivity and community size (cf. (17)) are $w_1 = 1$ and $w_2 = 2$, respectively. The anomaly detection threshold ξ is 0.6 for Scenarios 1 and 6 and 0.8 for the other scenarios.

Table I shows the comparison of our method with the BotHunter method in the five scenarios. In general, our method has lower precision but higher recall. The recall of BotHunter is very low, which is likely due to two reasons: (1) the data was collected in the gateway of an intranet and the bots inside the intranet may have been infected before the dataset was collected, and (2) information from external bots is not directly observed and is not sufficient for dialog analysis. In contrast, our method tends to be more aggressive in reporting alarms since our botnet discovery is based on modularity-based community detection, which is likely to detect large communities. This is also the reason why we add a regularization term for community size in (17) to alleviate this tendency, and we can increase the weight w_2 to achieve higher granularity. In botnet detection, recall is more important than precision because missing infected machines may cause serious security issues while false alarms are more tolerable.

There are several ways of combining precision and recall to give an overall metric for algorithm performance. Table I lists two popular metrics: F1-measure and G-measure [24]. The F1-measure is the harmonic mean of precision and recall and the G-measure is the geometric mean of precision and recall. In Scenario 8, BotHunter fails to report any bot IP. For all other scenarios, our

2325-5870 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2532804, IEEE Transactions on Control of Network Systems

Table I
SUMMARY OF RESULTS ON THE CTU DATASET.

Scenario	**Method**	**tTP**	**tTN**	**tFP**	**tFN**	**FPR**	**Recall (TPR)**	**Precision**	**F1 Measure**	**G Measure**
1	Our Method	7.52	66.81	1.19	90.50	0.017	0.077	0.86	**0.14**	**0.26**
	BotHunter	1.59	73.80	0.18	109	0.0024	0.014	0.90	0.028	0.11
2	Our Method	3.77	44.03	0.97	77.64	0.022	0.046	0.80	**0.088**	**0.19**
	BotHunter	1.65	46.9	0.05	75	0.0011	0.022	0.97	0.042	0.14
6	Our Method	4.05	15.18	1.82	28.66	0.11	0.12	0.69	**0.21**	**0.29**
	BotHunter	2.53	20.9	0.02	37.3	0.00096	0.064	0.99	0.12	0.25
8	Our Method	13.42	219.0	9.010	292.00	0.040	0.044	0.60	**0.082**	**0.16**
	BotHunter	0	229	0.11	309	0.00048	0	0	-	0
9	Our Method	5.21	56.04	1.97	59.66	0.034	0.080	0.73	**0.14**	**0.24**
	BotHunter	1.76	57.9	0.06	86.9	0.001	0.02	0.97	0.039	0.14

method has better F1-measure and G-measure than the BotHunter method.

D. Random graph model selection in mixed models

This section presents results of our random graph model selection algorithm on a mixed model. It also presents a way to determine the penalty coefficient θ (cf. Eq. (10)) through experiments. We first define the following random graph model. To generate a graph $\mathcal{G}_{mix} = (\mathcal{V}_{mix}, \mathcal{E}_{mix})$ with this model, we partition $\mathcal{V}_{mix}$ into two subsets $\mathcal{V}_{ER}$ and $\mathcal{V}_{PA}$. Then, graphs $\mathcal{G}_{ER} = (\mathcal{V}_{ER}, \mathcal{E}_{ER})$ and $\mathcal{G}_{PA} = (\mathcal{V}_{PA}, \mathcal{E}_{PA})$ are generated by the ER model and the PA model, respectively. The edge set of $\mathcal{G}_{mix}$ is $\mathcal{E}_{mix} = \mathcal{E}_{ER} \cup \mathcal{E}_{PA}$.

We let $|\mathcal{V}_{ER}| = |\mathcal{V}_{PA}|$ and generate a sequence of graphs using the random graph model described above. Taking some samples of node degrees $\mathscr{D} = \{d_1, \ldots, d_{|\mathscr{D}|}\}$ from the graphs, we can assume that the maximum likelihoods of $\mathscr{D}$ for the ER and the PA models are equal, namely, $\max_\lambda L_{ER}(\mathscr{D}; \lambda) = \max_\gamma L_{PA}(\mathscr{D}; \gamma)$. It follows from (10)

$$\theta \sum_i 1(d_i = 0) = \max_\gamma L^+_{PA}(\mathscr{D}; \gamma) - \max_\lambda L_{ER}(\mathscr{D}; \lambda). \tag{22}$$

We can use (22) to estimate θ approximately using least-squares. This approach generalizes to real-world datasets. We can ask experts to give estimates of $\max_\gamma L_{PA}(\mathscr{D}; \gamma) - \max_\lambda L_{ER}(\mathscr{D}; \lambda)$ for each $\mathscr{D}$ and $\sum_i 1(d_i = 0)$ is available from the sample $\mathscr{D}$. Again, we can use least-squares to estimate θ.

E. Description of CAIDA dataset

Next we create a dataset by mixing real-world botnet traffic with real-world background traffic. For the botnet traffic, we use the CAIDA "DDoS Attack 2007" dataset [26]. It includes traces from a Distributed Denial-of-Service (DDoS) attack on August 4, 2007. The DDoS attack attempts to block access to the targeted server by consuming computing resources on the server and all of the bandwidth connecting the server to the Internet.

The total size of the dataset is 21 GB covering about one hour (20:50:08 UTC to 21:56:16 UTC). This dataset only contains attacking traffic to the victim; all other traffic, including the C&C traffic, has been removed. The dataset consists of two parts. The first part is the traffic when the botnet initiates the attack (between 20:50 UTC and 21:13 UTC). In this stage, the bots probe whether they can reach the victim. The amount of traffic from the botnet during this period is small, thus, it is very challenging to detect it using only network load. The second part is the attack traffic which starts around 21:13 UTC when the network load increases rapidly (within a few minutes) from about 100 Kb/s to about 80 Mb/s. Although the DDoS attack itself (after 21:13 UTC) is trivial to detect, we apply our approach to a 5-minute segment between 20:50 UTC and 21:13 UTC. This is traffic from the pre-attack stage when the botnet probes the target. Botnet traffic during this period is low-intensity (about 100 Kb/s). A successful detection during this stage provides network administrators enough time (about 20 min) to take preventive actions, such as, blocking suspected bot IPs.

For the background traffic, we use trace 6 in the University of Twente traffic traces data repository (`simpleweb`) [27]. This trace was measured in a 100 Mb/s Ethernet link connecting a university unit to the Internet. This is a relatively small unit with around 35 employees and a little over 100 students. There are 100 workstations at this location which all have a 100 Mb/s LAN connection. The core network consists of a 1 Gb/s connection. The recordings took place between the external optical fiber modem and the first firewall. The measured link was only mildly loaded during this period. The background traffic we choose lasts for $3,600$ seconds. The botnet traffic is mixed with background traffic between $2,000$ and $2,300$ seconds.

F. Results of network anomaly detection

We first divide the background traffic into 10-second windows and create a sequence $\mathscr{G}_{bgrnd}$ of 360 back-

2325-5870 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2532804, IEEE Transactions on Control of Network Systems

ground *interaction graphs*. We apply the model selection technique described in Section III-B1 to $\mathscr{G}_{bgrnd}$.

We first randomly sample 50 *interaction graphs* from $\mathscr{G}_{bgrnd}$ and sample the degrees of 20 nodes in each *interaction graph* uniformly. As a result, our total sample size is $|\mathscr{D}| = 1,000$. In our experiment, we set $\theta = 2.44$. The MLE of the PA model is $\hat{\gamma} = 1.82$ and its log likelihood $L_{PA}(\mathscr{D}; \hat{\gamma}) = -3,072.58$. The MLE of the ER model is $\hat{\lambda} = 0.025$ and its log-likelihood $L_{ER}(\mathscr{D}; \hat{\lambda}) = -643.78$. We use a threshold $\eta = 0$ in applying the GLRT and assume no prior knowledge of the model. GLRT selects the ER model (cf. (11)).

Fig. 2-A shows the detection results. The blue "+" markers correspond to $I_{ER}(\boldsymbol{\mu}_i; \hat{\lambda})$ for each window i, $i = 1, \ldots, 360$, where $\boldsymbol{\mu}_i$ is the empirical degree distribution of *interaction graph* i. The (red) dashed line shows the threshold $\xi = 0.18$. There are 36 abnormal *interaction graphs*, namely, $|\mathcal{A}| = 36$. There are 30 *interaction graphs* that have botnet traffic and 29 *interaction graphs* are correctly identified. The *interaction graph* corresponding to the time range $[2,000, 2,010]$ is missed. Being the start of the botnet traffic, this range has very low botnet activity, which may explain the miss-detection. In addition, there are two groups of false alarms—3 false alarms around 3,000 s and 4 false alarms around 3,500 s. Fig. 2-B shows the Receiver Operating Characteristic (ROC) curve of the network anomaly detection stage.

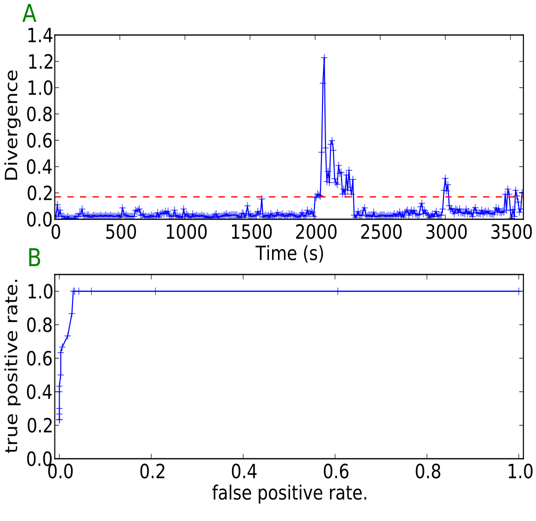

Figure 2. **(A)**: the rate function value $I_{ER}(\boldsymbol{\mu}_i; \hat{\lambda})$ for each window i. The x-axis shows the start time of each window. The background traffic lasts for 3,600 seconds and the botnet traffic is added between 2,000 and 2,300 seconds. **(B)**: the ROC curve. The x-axis plots the false positive rate and the y-axis plots the true positive rate.

G. Results of botnet discovery

The *botnet discovery* stage aims to identify bots based on the information in $\mathcal{A}$. The first step is to identify a set of pivotal nodes $\mathcal{N}$ (cf. Def. 4 and Eq. (14)). Let e_{max} be the maximum *total interaction measure* of all nodes and $\mathscr{S}_e^{Norm} = \{e_i/e_{max} : i = 1, \ldots n\}$ the normalized set of *total interaction measures*. Fig. 3 plots $\mathscr{S}_e^{Norm}$ in descending order using log-scale for the y-axis. Each blue "+" marker represents one node. The blue curve in Fig. 3, being quite steep, clearly indicates the existence of influential pivotal nodes. The red dashed line in Fig. 3 plots the chosen threshold τ used in Def. 4, which results in 3 pivotal nodes. Only one pivotal node (the one with maximum *total interaction measure*) belongs to the botnet. The other two pivotal nodes are active normal nodes. These two falsely detected pivotal nodes correspond to the two false-alarm groups described in Section V-F.

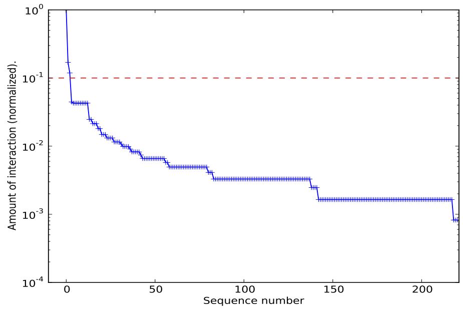

Figure 3. Sorted amount of interaction in $\mathcal{A}$ defined by (14). y-axis is in log-scale. The red dashed line plots the threshold τ we used.

Our dataset has 396 nodes, including 136 bots and 260 normal nodes. Among the 396 nodes, only 213 nodes have positive sample standard deviations $\sigma(X_i)$. Let $\mathcal{V}_p = \{i : \sigma(X_i) > 0\}$ be the set of these nodes. Fig. 4 plots the correlation matrix of X_i, X_j for all $i, j \in \mathcal{V}_p$. We can easily observe two correlation groups.

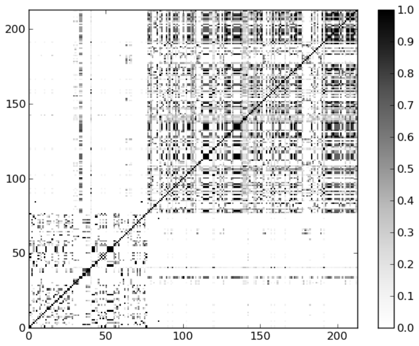

Figure 4. The correlation matrix of X_i, X_j for all $i, j \in \mathcal{V}_p$ (plotted by the `pcolor` command in the `pylab` python package).

We calculate the SCG $\mathcal{C}$ using Def. 5 and threshold $\tau_p = 0.3$. In $\mathcal{C}$, there are 191 isolated nodes with degree

2325-5870 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

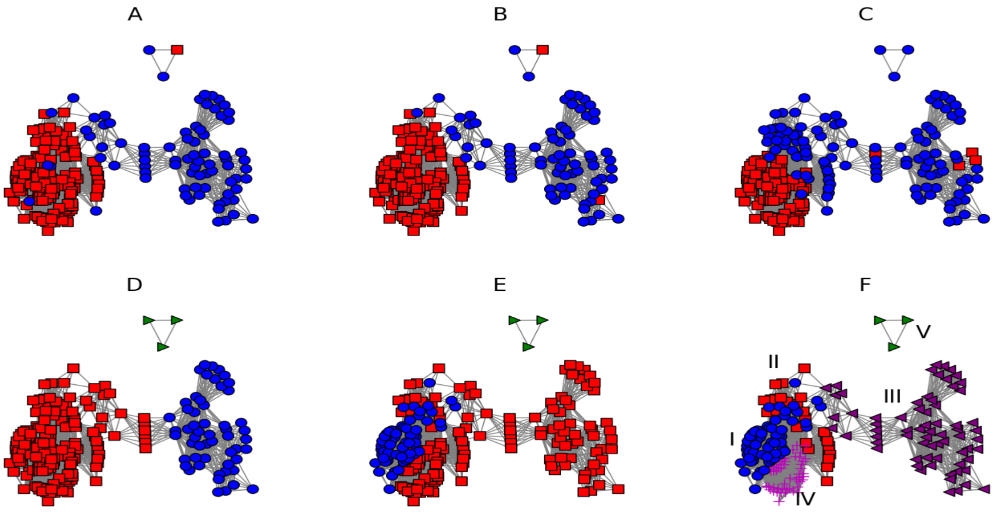

Figure 5. Comparison of different community detection techniques on the SCG. In A–C, red squares are bots and blue circles are normal nodes. In D–F, red squares indicate the group with the highest average *pivotal interaction measure*, while blue circles indicate the group with the lowest one. **(A)**: ground-truth communities of bots and normal nodes. **(B)**: result of our botnet discovery approach. **(C)**: result of the vector programming method in [21]. **(D)**: result of the walktrap method [28] with three communities. **(E)**: result of the leading eigenvector method [19] with 3 communities. **(F)**: result of the leading eigenvector method with five clusters.

zero. The subgraph formed by the remaining 205 nodes has two connected components (Fig. 5-A). Fig. 5-A plots normal nodes as blue circles and bots as red squares. Although the bots and the normal nodes clearly belong to different communities, the two communities are not separated in the narrowest part. Instead, the separating line is closer to the bots.

We apply our *botnet discovery* method to $\mathcal{C}$. The result (Fig. 5-B) is very close to the ground truth (Fig. 5-A). As comparison, we also apply other community-detection methods to the 205-node subgraph.

The first method is the vector programming method proposed in [21], which is a special case of our method using $w_1 = 0$ and $w_2 = 0$ in (17). This approach, however, misses some bots (5-C).

The second method is the `walktrap` method in [28], which defines a distance metric between nodes based on a random walk and applies hierarchical clustering. When the desirable number of communities, a required parameter, equals to two, the method reports the two connected components – a reasonable, yet uninformative result for botnet discovery. To make the results more meaningful, we use `walktrap` to find three communities and ignore the smallest one that corresponds to the smaller connected component (right triangles in Fig. 5-D). The community with higher mean of the pivotal interaction measure is detected as the botnet, and the remaining nodes are called normal. The `walktrap` method separates bots and normal nodes in the narrowest part of the graph, a reasonable result from the perspective of community detection (Fig. 5-D). However, a comparison with the ground-truth reveals that a lot of normal nodes are falsely reported as bots.

The third method is Newman's leading eigenvector method [19], a classical modularity-based community detection method. This method first calculates the eigenvector corresponding to the second-largest eigenvalue of the modularity matrix $\mathbf{M}$, namely the *leading eigenvector*. The solution $\mathbf{s} = (s_1, \ldots, s_n)$ is then constructed by letting s_i be the sign of the ith element of the *leading eigenvector*. The method can be generalized for detecting multi-communities [19]. Similar to the `walktrap` method, the leading eigenvector method reports two connected components when the desirable community number is two. We also use this method to find three communities and ignore the smallest one. Again, the community with higher mean of *pivotal interaction measure* is detected as the botnet.

Different from the previous methods, the eigenvector method makes a completely wrong prediction of the botnet. The community whose majority are bots (blue circles in Fig. 5-E) is wrongly detected as the normal part and the community formed by the remaining nodes is wrongly detected as the botnet. Despite being part of the real botnet, the community of blue circles in Fig. 5-E actually has lower mean of *pivotal interaction measure*, i.e., less overall communication with pivotal nodes.

After dividing the SCG $\mathcal{C}$ into five communities using the leading vector approach for multi-communities [19], we observe that the botnet itself is heterogeneous and

2325-5870 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2532804, IEEE Transactions on Control of Network Systems

divided into three groups. Both the group with the highest mean of *pivotal interaction measure* (Group II in Fig. 5-F) and the group with the lowest mean (Group I in Fig. 5-F) are part of the botnet.

Because of this heterogeneity, some groups of the botnet may be misclassified. The leading vector method wrongly separates Group I from the rest as a single community, and merges Group II & IV with the normal nodes (Group III). Because Group I has the lowest *pivotal interaction measure*, it is wrongly detected as normal, causing Groups II, III, IV to be detected as the botnet. Similarly, the vector programming method wrongly detects a lot of nodes in Group II, which should be bots, as normal nodes.

By taking the *pivotal interaction measure* into consideration, the misclassification can be avoided. In our formulation of the refined modularity (17), the term $w_1 \sum_i r_i s_i$ maximizes the difference of the *pivotal interaction measure* of the botnet and that of the normal part. Owing to this term, our method makes few mistakes for nodes in Group II since they have high pivotal interaction measures. Let $\mathscr{S}_r^- = \{r_i : s_i = -1\}$ and $\mathscr{S}_r^+ = \{r_i : s_i = 1\}$ be the set of pivotal interaction measures for normal nodes and bots. Table II shows the mean of $\mathscr{S}_r^-$ (Col. 1) and $\mathscr{S}_r^+$ (Col. 2) for ground truth (Row 1), our method (Row 2), the `walktrap` method (Row 3), and the leading eigenvector method with 3 communities (3-LEV, Row 4), respectively. The difference between the mean of $\mathscr{S}_r^+$ and $\mathscr{S}_r^-$ (Col. 3 of Table II) of 3-LEV, whose result is unreasonable, is significantly smaller than the rest of the methods. In comparison, the difference in our method is much larger and closer to the ground truth.

Table II

Statistics on the pivotal interaction measures

	Normal Mean	**Bot Mean**	**Difference**
Ground Truth	0.0021	0.024	0.0219
Our Method	0.0017	0.024	0.0223
Walktrap	0.0019	0.021	0.0191
3-LEV	0.011	0.018	0.0078

VI. Conclusions

In this paper, we propose a novel method of botnet detection that consists of two stages. The first stage applies a sliding window to network traffic and monitors anomalies in the network. We propose two anomaly detection methods, both of which are based on large deviations results, for flow and packet level data, respectively. For both anomaly detection methods, an anomaly can be represented as a set of *interaction records*.

Once instances of anomalies have been identified, we proposed a method for detecting the compromised nodes. This is based on ideas from community detection in social networks. However, we devised a refined modularity measure that is suitable for botnet detection. The refined modularity also addresses some limitations of *modularity* by adding regularization terms and combining information of *pivotal interaction measure* and SCGs.

Appendix A
LDP for the degree distribution in the PA model

In the generalized Polya urn model, let $h_i^n(j)$ denote the number of urns with i balls at time j. For $0 \le j \le n$ and $d > 0$, define $\mathbf{h}^{n,d}(j) = (h_0^n(j), \ldots, h_d^n(j), \bar{h}_{d+1}^n(j))$ to be the d-truncated degree distribution at time j, where $\bar{h}_{d+1}^n(j) = \sum_{k \ge d+1} h_k^n(j)$. $\mathscr{H}^{n,d} = \left\{\mathbf{h}^{n,d}(0), \ldots, \mathbf{h}^{n,d}(n)\right\}$ is a Markov chain with initial state $\mathbf{h}^{n,d}(0)$ corresponding to the initial urn configuration $\mathcal{U}_0$. We interpolate $\mathscr{H}^{n,d}/n$ into a continuous process $\mathscr{X}^{n,d} = \left\{\mathbf{x}^{n,d}(t) : 0 \le t \le 1\right\}$ as:

$$\mathbf{x}^{n,d}(t) = (1/n)\mathbf{h}^{n,d}(\lfloor nt \rfloor) \\ + (t - \lfloor nt \rfloor/n)\left(\mathbf{h}^{n,d}(\lfloor nt \rfloor + 1) - \mathbf{h}^{n,d}(\lfloor nt \rfloor)\right).$$

We can extend this definition into an infinite-dimensional process $\mathscr{X}^{n,\infty} = \{\mathbf{x}^{n,\infty}(t) : 0 \le t \le 1\}$ by removing the truncation.

Suppose we can observe an empirical d-truncated degree distribution $\boldsymbol{\varphi}(t) = (\varphi_0(t), \ldots, \varphi_d(t), \bar{\varphi}_{d+1}(t))$, where $\bar{\varphi}_{d+1}(t) = 1 - \sum_{k=0}^{d} \varphi_k(t)$. Define $\dot{\boldsymbol{\varphi}}(t) = (\dot{\varphi}_0(t), \ldots, \dot{\varphi}_d(t), \dot{\bar{\varphi}}_{d+1}(t))$ as the time derivative of $\boldsymbol{\varphi}(t)$. For $d \ge 0$, [13] establishes a sample-path LDP for $\mathscr{X}^{n,d}$ with rate function (recall the $[\cdot]_i$ notation; Sec. I)

$$I_d(\boldsymbol{\varphi}) = \int_0^1 \left[(1 - [\dot{\boldsymbol{\varphi}}(t)]_0) \log \frac{1 - [\dot{\boldsymbol{\varphi}}(t)]_0}{p(t) + (1 - p(t)) \frac{\beta(t)\varphi_0(t)}{(1+\beta(t))t}} \right. \\ + \sum_{i=1}^{d} (1 - [\dot{\boldsymbol{\varphi}}(t)]_i) \log \frac{1 - [\dot{\boldsymbol{\varphi}}(t)]_i}{(1 - p(t)) \frac{(i+\beta(t))\varphi_i(t)}{(1+\beta(t))t}} \\ + \left(1 - \textstyle\sum_{i=0}^{d} (1 - [\dot{\boldsymbol{\varphi}}(t)]_i)\right) \times \\ \left. \log \frac{1 - \sum_{i=0}^{d} (1 - [\dot{\boldsymbol{\varphi}}(t)]_i)}{(1 - p(t)) \left(1 - \frac{\sum_{i=0}^{d} (i+\beta(t))\varphi_i(t)}{(1+\beta(t))t}\right)} \right] dt. \tag{23}$$

Similarly, $\mathscr{X}^{n,\infty}$ satisfies a sample-path LDP with rate function

$$I_\infty(\boldsymbol{\varphi}) = \lim_{d \to \infty} I_d(\boldsymbol{\varphi}). \tag{24}$$

In applying this result, we assume that the observed network is generated by a process that evolves with constant rate, i.e., $\boldsymbol{\varphi}(t) = \boldsymbol{\mu} t$. It follows $\dot{\boldsymbol{\varphi}}(t) = \boldsymbol{\mu}$. The rate functions in (4) and (5) are special cases of (24).

2325-5870 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2532804, IEEE Transactions on Control of Network Systems

Appendix B
Rate function for a model that selects between the offset BA and the CHJ

We give an outline of the key argument. Suppose we build a random graph by selecting with probability π_{BA} the offset BA process and with probability $\pi_{CHJ} = 1 - \pi_{BA}$ the CHJ process. Recall our definition of the d-truncated degree distribution process $\mathscr{X}^{n,d} = \{\mathbf{x}^{n,d}(t) : 0 \le t \le 1\}$ and let $\mathscr{X}^{n,d}_{PA}$, $\mathscr{X}^{n,d}_{BA}$, and $\mathscr{X}^{n,d}_{CHJ}$ be the corresponding processes for the generic PA model, the offset BA model, or the CHJ model, respectively. For any closed set $\mathcal{B}$ of trajectories

$$P(\mathscr{X}^{n,d}_{PA} \in \mathcal{B}) \le P(\mathscr{X}^{n,d}_{BA} \in \mathcal{B}) + P(\mathscr{X}^{n,d}_{CHJ} \in \mathcal{B}),$$

and since the two terms on the rhs satisfy LDPs with rate functions $I_{BA}(\boldsymbol{\mu};\alpha)$ and $I_{CHJ}(\boldsymbol{\mu};p)$ respectively, we obtain

$$\limsup_{n\to\infty} \frac{1}{n} \log P(\mathscr{X}^{n,d}_{PA} \in \mathcal{B}) \le \inf_{\{\boldsymbol{\mu}: \mathscr{X}^{n,d}(\boldsymbol{\mu}) \in \mathcal{B}\}} I_{PA}(\boldsymbol{\mu}, \gamma), \tag{25}$$

where $\mathscr{X}^{n,d}(\boldsymbol{\mu})$ are trajectories that evolve at a constant rate $\boldsymbol{\mu}$.

To establish a lower bound note that

$$P(\mathscr{X}^{n,d}_{PA} \in \mathcal{B}) \ge P(\mathscr{X}^{n,d}_{BA} \in \mathcal{B})$$
$$\text{and } P(\mathscr{X}^{n,d}_{PA} \in \mathcal{B}) \ge P(\mathscr{X}^{n,d}_{CHJ} \in \mathcal{B}).$$

It follows that the LDP lower bound for $P(\mathscr{X}^{n,d}_{PA} \in \mathcal{B})$ holds with rate function $I_{PA}(\boldsymbol{\mu}, \gamma)$.

Appendix C
Proof of Theorem IV.1

Proof: Letting $\mathbf{x} = (\mathbf{s}, 1)$, problem (18) becomes:

$$\begin{aligned} f^* = \max \quad & \mathbf{x}'\mathbf{W}\mathbf{x} \\ \text{s.t.} \quad & x_i^2 = 1, \qquad i = 1, \ldots, n+1. \end{aligned} \tag{26}$$

$\mathbf{W}$ is a $(n+1) \times (n+1)$ symmetric matrix, not necessarily $\succeq 0$, yet, its smallest eigenvalue χ^-_{min} is real. Eq. (20) is an SDP relaxation of (26). Suppose now $\mathbf{P} = \mathbf{W} + \varrho\mathbf{I}$ is a regularized matrix. $\mathbf{P} \succeq 0$ if

$$\varrho \ge -\chi^-_{\text{min}}. \tag{27}$$

Consider the regularized problem

$$\begin{aligned} f^*_{\text{reg}} = \max \quad & \mathbf{x}'\mathbf{P}\mathbf{x} \\ \text{s.t.} \quad & x_i^2 = 1, \qquad i = 1, \ldots, n+1. \end{aligned} \tag{28}$$

Note that the regularized term $\varrho\mathbf{x}'\mathbf{I}\mathbf{x}$ is a constant for any feasible solution because $\mathbf{x}'\mathbf{I}\mathbf{x} = \sum_i x_i^2 = n+1$. As a result, (26) and (28) should have the same optimal solution $\mathbf{x}^*$ but different objective values. Then,

$$f^*_{\text{reg}} = f^* + \varrho(n+1). \tag{29}$$

By setting $\mathbf{X} = \mathbf{x}\mathbf{x}'$ the SDP relaxation of (28) is:

$$\begin{aligned} \max \quad & \text{Tr}(\mathbf{X}\mathbf{P}) \\ \text{s.t.} \quad & \mathbf{X} \succeq 0, \\ & X_{ii} = 1, \qquad i = 1, \ldots, n+1. \end{aligned} \tag{30}$$

Suppose $\mathbf{X}^*$ is an optimal solution of (30); then

$$\frac{2}{\pi} \text{Tr}\left(\mathbf{X}^*\mathbf{P}\right) \le f^*_{\text{reg}} \le \text{Tr}\left(\mathbf{X}^*\mathbf{P}\right) \tag{31}$$

if (27) is satisfied. The upper bound is due to the relaxation and the lower bound follows from Nesterov's extension to the MAXCUT bound [11]. Moreover,

$$\begin{aligned} \text{Tr}\left(\mathbf{X}^*\mathbf{P}\right) &= \text{Tr}\left(\mathbf{X}^*\mathbf{W}\right) + \varrho \text{Tr}\left(\mathbf{X}^*\right) \\ &= f^*_{\text{relax}} + \varrho\,(n+1)\,. \end{aligned} \tag{32}$$

Combining (29), (31), and (32), we have

$$\frac{2}{\pi}(f^*_{\text{relax}} + \varrho(n+1)) \le f^* + \varrho(n+1) \tag{33}$$

for $\varrho \ge -\chi^-_{\text{min}}$. The lower bound in Th. IV.1 can be proved by letting ϱ take its minimum value $-\chi^-_{\text{min}}$ and reorganizing the terms in (33). The upper bound in Th. IV.1 is a standard property of SDP relaxation. ■

References

[1] "DDoS Protection Whitepaper," 2012, http://www.neustar.biz/enterprise/resources/ddos-protection/ddos-attacks-survey-whitepaper#.UtwNR7Uo70o.

[2] W. T. Strayer, R. Walsh, C. Livadas, and D. Lapsley, "Detecting botnets with tight command and control," in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on.* IEEE, 2006, pp. 195–202.

[3] G. Gu, J. Zhang, and W. Lee, "BotSniffer: detecting botnet command and control channels in network traffic," in *Proceedings of the 15th Annual Network and Distributed System Security Symposium*, 2008.

[4] G. Stringhini, T. Holz, B. Stone-Gross, C. Kruegel, and G. Vigna, "Botmagnifier: Locating spambots on the internet." in *USENIX Security Symposium*, 2011.

[5] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, and W. Lee, "Bothunter: Detecting malware infection through ids-driven dialog correlation." in *Usenix Security*, vol. 7, 2007, pp. 1–16.

[6] A. Dembo and O. Zeitouni, *Large Deviations Techniques and Applications*, 2nd ed. NY: Springer-Verlag, 1998.

[7] I. C. Paschalidis and G. Smaragdakis, "Spatio-temporal network anomaly detection by assessing deviations of empirical measures," *IEEE/ACM Trans. Networking*, vol. 17, no. 3, pp. 685–697, 2009.

[8] J. Wang and I. C. Paschalidis, "Statistical traffic anomaly detection in time-varying communication networks," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 2, pp. 100–111, 2015.

[9] J. Wang, D. Rossell, C. G. Cassandras, and I. C. Paschalidis, "Network anomaly detection: A survey and comparative analysis of stochastic and deterministic methods," in *Proceedings of the 52nd IEEE Conference on Decision and Control*, Florence, Italy, December 2013, pp. 182–187.

[10] J. Wang and I. C. Paschalidis, "Botnet detection using social graph analysis," in *52nd Annual Allerton Conference on Communication, Control, and Computing*, Monticello, Illinois, October 2014.

[11] A. Ben-Tal and A. Nemirovski, *Lectures on Modern Convex Optimization.* SIAM, 2001.

2325-5870 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCNS.2016.2532804, IEEE Transactions on Control of Network Systems

[12] S. Mukherjee, “Large deviation for the empirical degree distribution of an Erdős-Rényi graph,” *arXiv preprint arXiv:1310.4160*, pp. 1–23, 2013.

[13] J. Choi and S. Sethuraman, “Large deviations for the degree structure in preferential attachment schemes,” *The Annals of Applied Probability*, vol. 23, no. 2, pp. 722–763, 2013.

[14] A. Collevecchio, C. Cotar, and M. LiCalzi, “On a preferential attachment and generalized Polya’s urn model,” *The Annals of Applied Probability*, vol. 23, no. 3, pp. 1219–1253, Jun. 2013.

[15] F. Chung, S. Handjani, and D. Jungreis, “Generalizations of Polya’s urn Problem,” *Annals of Combinatorics*, vol. 7, no. 2, pp. 141–153, Jul. 2003.

[16] I. C. Paschalidis and D. Guo, “Robust and distributed stochastic localization in sensor networks: Theory and experimental results,” *ACM Trans. Sensor Networks*, vol. 5, no. 4, pp. 34:1–34:22, 2009.

[17] D. Shah and T. Zaman, “Community detection in networks: The leader-follower algorithm,” *arXiv preprint arXiv:1011.0774*, pp. 1–13, 2010. [Online]. Available: http://arxiv.org/abs/1011.0774

[18] M. Newman, “Detecting community structure in networks,” *The European Physical Journal B - Condensed Matter*, vol. 38, no. 2, pp. 321–330, Mar. 2004.

[19] ——, “Finding community structure in networks using the eigenvectors of matrices,” *Physical review E*, 2006.

[20] S. Fortunato and M. Barthelemy, “Resolution limit in community detection,” *Proceedings of the National Academy of Sciences of the United States of America*, pp. 1–8, 2007.

[21] G. Agarwal and D. Kempe, “Modularity-maximizing graph communities via mathematical programming,” *The European Physical Journal B*, vol. 66, no. 3, pp. 409–418, Nov. 2008.

[22] J. Duch and A. Arenas, “Community detection in complex networks using extremal optimization,” *Physical review E*, no. 2, pp. 2–5, 2005.

[23] A. D’Aspremont and S. Boyd, “Relaxations and randomized methods for nonconvex QCQPs,” *EE392o Class Notes, Stanford University*, no. 1, pp. 1–16, 2003.

[24] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, “An empirical comparison of botnet detection methods,” *Computers & Security*, vol. 45, pp. 100–123, 2014.

[25] J. Wang, J. Zhang, and I. C. Paschalidis, “General Anomaly Detector (GAD),” https://github.com/hbhzwj/GAD, 2014.

[26] “The CAIDA UCSD ”DDoS Attack 2007” Dataset,” CAIDA, 2013, http://www.caida.org/data/passive/ddos-20070804_dataset.xml.

[27] R. R. R. Barbosa, R. Sadre, A. Pras, and R. van de Meent, “Simpleweb/university of Twente traffic traces data repository,” http://eprints.eemcs.utwente.nl/17829/, Technical Report TR-CTIT-10-19, April 2010.

[28] P. Pons and M. Latapy, “Computing communities in large networks using random walks,” *Computer and Information Sciences-ISCIS*, 2005.

Jing (Conan) Wang received the Ph.D. degree in Systems Engineering from Boston University, Boston, MA, USA in 2015, and the B.E. degree in Electrical and Information Engineering from Huazhong University of Science and Technology, China, in 2010. He is now a software engineer at Google Inc. His research interests include interest-based advertising, cyber security, and approximate dynamic programming.

Ioannis Ch. Paschalidis (M’96–SM’06–F’14) received the M.S. and Ph.D. degrees both in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 1993 and 1996, respectively. In September 1996 he joined Boston University where he has been ever since. He is a Professor and Distinguished Faculty Fellow at Boston University with appointments in the Department of Electrical and Computer Engineering, the Division of Systems Engineering, and the Department of Biomedical Engineering. He is the Director of the Center for Information and Systems Engineering (CISE). He has held visiting appointments with MIT and Columbia University, New York, NY, USA. His current research interests lie in the fields of systems and control, networking, applied probability, optimization, operations research, computational biology, and medical informatics.

Dr. Paschalidis is a recipient of the NSF CAREER award (2000), several best paper and best algorithmic performance awards, and a 2014 IBM/IEEE Smarter Planet Challenge Award. He was an invited participant at the 2002 Frontiers of Engineering Symposium, organized by the U.S. National Academy of Engineering and the 2014 U.S. National Academies Keck Futures Initiative (NAFKI) Conference. He is the inaugural Editor-in-Chief of the IEEE Transactions on Control of Network Systems.

2325-5870 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.