Detecting Malicious Clients in ISP Networks Using HTTP Connectivity Graph and Flow Information

Lei Liu†, Sabyasachi Saha*, Ruben Torres*, Jianpeng Xu†,Pang-Ning Tan†, Antonio Nucci*, Marco Mellia‡
†Dept. of Computer Science, Michigan State University, East Lansing, MI 48824
*Narus Inc. 570 Maude Court, Sunnyvale, CA 94085
‡Politecnico di Torino, Italy
Email: †{liulei1, xujianpe, ptan}@msu.edu, *{ssaha, rtorres, anucci}@narus.com, ‡ mellia@tlc.polito.it

***Abstract*—This paper considers an approach to identify previously undetected malicious clients in Internet Service Provider (ISP) networks by combining flow classification with a graph-based score propagation method. Our approach represents all HTTP communications between clients and servers as a weighted, near-bipartite graph, where the nodes correspond to the IP addresses of clients and servers while the links are their interconnections, weighted according to the output of a flow-based classifier. We employ a two-phase alternating score propagation algorithm on the graph to identify suspicious clients in a monitored network. Using a symmetrized weighted adjacency matrix as its input, we show that our score propagation algorithm is less vulnerable towards inflating the malicious scores of popular Web servers with high in-degrees compared to the normalization used in PageRank, a widely used graph-based method. Experimental results on a 4-hour network trace collected by a large Internet service provider showed that incorporating flow information into score propagation significantly improves the precision of the algorithm.**

I. Introduction

Malicious attacks on the Internet have been on the rise for the last few years. According to Symantec's Internet Security Threat Report [1], the number of Web attacks in 2012 has increased by 42% compared to its previous year. Sophisticated botnets have been widely used in these attacks to coordinate spam campaigns, launch denial-of-service attacks, or steal sensitive information. Increasingly, the bot activities are coordinated via sophisticated and stealthy command-and-control (C&C) channels designed to evade detection by traditional signature-based Intrusion Detection and Prevention Systems (IPS/IDS). Detection of such C&C channels is difficult for many reasons, including the use of: (i) HTTP protocols to bypass firewalls, (ii) encryption to obscure payloads, and (iii) "domain fast-flux" to constantly change locations of the command and control servers. Despite the various evasion techniques employed by the botnets, one unavoidable aspect they cannot hide is that the infected clients or bots must communicate back to their C&C servers. In this regard, we utilize the IP-address connectivity graph ("who talks to whom") to detect the botnets.

Many recent approaches to identify Web attacks have focused on detecting malicious URLs in general [2], [3], [4] or more specific threats such as phishing [5], drive-by-downloads [6] and command and control communication [7]. These approaches rely mostly on analysis of the content of the downloaded files or the lexical features of URLs used in the communication. Since they do not consider the network communication graph, they often fall short in identifying other infected clients in the network, beyond those involved in sending HTTP requests for malicious content. Botnet detection methods that identify hosts engaging in command and control communication by correlating their sequences of alerts [8] or clustering their flow-level statistics [9] are also prone to such problem. Similarly, a signature-based or classifier-based IDS can only identify malicious clients associated with known malwares. Such approaches are unable to detect zero-day malware attacks for which the IDS has no knowledge of their threat signatures nor labeled examples.

In this paper, we seek to identify previously undetected malicious clients beyond those found by an IDS by analyzing the HTTP connections established by the clients in a monitored network. Our proposed approach leverages the benefits of both host-based and graph-based methods by combining the network communication graph, HTTP communication details, and information about the malicious clients detected by the IDS to identify additional undetected malicious clients in the network. First, we represent all the HTTP communication between the clients and servers as a directed graph, where the nodes correspond to clients and Web servers, and the links are directed from client to server nodes. Starting from an initial set of seed nodes that have been identified as malicious by a reputed, well-known commercial IDS, we apply a score propagation algorithm to transmit the score from the seeded nodes to other unlabeled nodes in the graph. The rationale here is that nodes that are linked to each other are more likely to have the same class label (malicious or non-malicious). Previously undetected nodes with high malicious score after propagation will be declared as *suspicious*.

While the idea of using score propagation algorithms for security-related problems is not new [10] [11], existing approaches, based on PageRank [12] and other variants of the random walk algorithm, consider a unipartite graph whose links are weighted either as a binary 0/1 value, or computed according to the similarities of their node attributes. Such approaches cannot be effectively applied to our problem for the following reasons. First, the HTTP communication network is a near-bipartite graph, where the majority of the nodes are either clients or Web servers, but not both. Due

ASONAM 2014, August 17-20, 2014, Beijing, China
978-1-4799-5877-1/14/$31.00 ©2014 IEEE

to the directed links in the graph, existing score propagation algorithms would propagate scores from clients to servers only, without considering the propagation from servers to clients. As a result, servers that are known to host malicious content will not be able to transmit their malicious scores to the clients that contact them during the random walk process. To overcome this problem, we design a two-phase alternating score-propagation method, which allows the clients and servers to independently propagate their scores through their outgoing and incoming links in different rounds.

Second, the standard approach of assigning a binary weight (1 if there is a connection and 0 otherwise) to each link—a strategy commonly used in spam detection [12], [10]—is insufficient to detect the Web attacks. For example, consider the graph shown in Figure 1(a) in which a malware infected client (as detected by an IDS) has contacted two Web servers. During score propagation, these two servers would receive the same score from the client irrespective of their type of flows. Suppose the traffic from C_1 to S_2 contains transmission of a known malicious file while the traffic from C_1 to S_1 involves an HTTP request to google.com. One would expect the malicious score propagated from C_1 to S_2 to be higher than that to S_1. A binary weighted graph will not distribute the score disproportionately, which makes S_1 equally suspicious as S_2. This example clearly illustrates the need to incorporate flow-level information into the score propagation algorithm. Since the flows are associated with the links instead of the nodes, a key challenge is to convert the flow information into a link weight that can be utilized by the score propagation algorithm. Using the node attribute similarity between two hosts [9] as link weight is also insufficient because a high similarity score does not necessarily imply that the score propagated should be high unless the flows between them were malicious. To overcome this challenge, we employ a flow-based classifier[1] to determine whether the flows through a link has any similarity with previously observed malicious flows and use this information to weigh the links between nodes. This strategy enables us to fuse the HTTP traffic flow classification results directly into the graph-based algorithm. For example, consider the graph shown in Figure 1(b), in which two infected clients, C_2 and C_3 contacted a malicious server S_3. Suppose both clients also contacted server S_4 and the link weights suggest some of their flows resemble previous malicious activities. The fact that S_4 is contacted only by infected clients would identify S_4 as suspicious. On the other hand, S_5, which was contacted by an infected and a non-infected client, would accumulate a much lower score as its link weights since there are no suspicious flows between the two nodes. This example demonstrates the rationale for combining communication graph with flow classification information.

Another potential limitation of using any random walk based algorithm is the score accumulation by high degree nodes. Since most high degree nodes correspond to popular Web servers such as google.com), this may inadvertently inflate their malicious scores during score propagation. Even if the flow-based classifier assigns a low (but non-zero) score to all the flows associated with the popular server, its overall malicious score is still high upon propagating and aggregating the scores from its neighbors. We show that the problem can be alleviated using a symmetrized weighted adjacency matrix constructed from the HTTP communication graph.

[1]This is a supervised classifier, trained offline on the previously IDS-labeled malicious flows to predict maliciousness score of the flows.

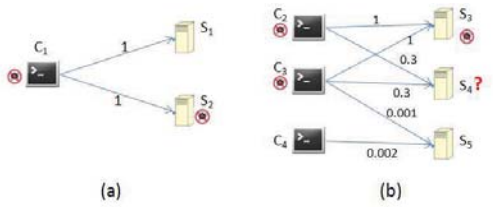

Fig. 1. A toy example illustrating the limitation of applying score propagation algorithm to a graph whose links have binary 0/1 weights.

The main contribution of our paper is the design of a methodology that combines HTTP connectivity graph with network traffic flow information to detect previously undetected malicious clients in a network. Unlike existing methods [9], [8], our algorithm utilizes the labels provided by an IDS and the outputs from a flow-based classifier to propagate the scores of known malicious clients and servers to other nodes in the network. We evaluated our approach using a 4-hour network trace from a large Commercial ISP. We showed that the proposed approach was able to identify previously undetected clients at higher precision and lower false alarm rates compared to standard PageRank-like algorithms.

II. Related Work

Most of the recent studies on Web-based attacks have focused on malicious URL detection. Ma et al. [2] combined lexical and host-level features to build statistical models for classifying malicious URLs, while Le et al. [5] relied only on the lexical features to detect phishing sites. Stokes et al. [4] leveraged the known malicious sites identified by an anti-malware software installed on the client machines to flag other Web pages that are linked to those sites. Zhang et al. [6] also employed anti-malware software on the client to generate regular expression of URLs suspected of distributing malware binaries. In contrast, our work focuses on identifying infected clients as opposed to malicious Web sites. The existing approaches may complement our work by providing the initial node labels or link weights for our flow-based score propagation algorithm.

Another related area of research is in botnet detection [8], [13], [9], [14], [15]. However, current approaches were limited to detecting botnets from hosts that were flagged by anomaly detection systems. They are not designed to identify malicious clients beyond those exhibiting anomalous network traffic activities. Graph-based methods have also been applied to other security-related problems such as fraud and spam detection[10]. Our work differs from previous graph-

based approaches for security-related problems in two ways. First, the graph constructed by our method is near-bipartite, corresponding to clients and servers participating in HTTP communication. In addition, our approach combines the outputs from an IDS and a flow-based classifier into the score propagation algorithm. We showed that the combination of HTTP connectivity and flow-based information outperforms the results using only link information.

III. Overview of System Architecture

In this section we provide an overview of the proposed system. We envision the system to be installed at the edge of an ISP or enterprise network. Our goal is to design a scalable system that detects infected clients inside the monitored network that were previously undetected by the IDS. However, this system is not for identifying such infected clients on the fly. We focus rather on the communication graph of the clients over a time period and analyze if some clients can be linked to the infected clients based on the sites they have communicated with. We use a score propagation algorithm to identify such clients.

Figure 2 shows the overall system architecture. It has six major components: (i) data capture module, (ii) intrusion detection/prevention system (IDS), (iii) feature extraction module, (iv) Flow classifier, (v) Graph construction and (vi) score propagation module. The data capture module is a Narus Semantic Traffic Analyzer (STA) to capture and parse the HTTP traffic flows needed for constructing the HTTP communication graph. An IDS is then used to classify the captured flows as malicious if their payload matches one of the pre-defined signatures. Otherwise, the unmatched flows are designated as "unknown". Flow classifier is a supervised classifier, discussed in SectionIV-B, computes malicious score of each network flow. Both the malicious and unknown flows are then provided to the graph construction module, which creates a weighted directed graph using the IP addresses of clients and servers as nodes and the flows between them as links. A more detailed discussion on the graph construction process is presented in Section IV-A.

The nodes are initially classified using labels provided by an IDS (or other available systems). Their initial threat levels are then propagated to neighboring nodes in the HTTP connectivity graph, where the propagated scores depend on both the degrees of the nodes as well as the malicious level of their flow information. The propagation is repeated until a maximum number of iterations is reached[2]. Nodes with high malicious score after propagation will be declared as *malicious*.

System components (i) to (iv) operate on each flow. The flow-classifier is trained offline and used to predict the maliciousness of each flow. The processes of graph construction and score propagation are also done offline from the stored data and can be highly parallelized. The graph is updated periodically and assumed to be static within a time period up to its next update.

[2]For our experiments, we found that the algorithm converges in less than 20 iterations.

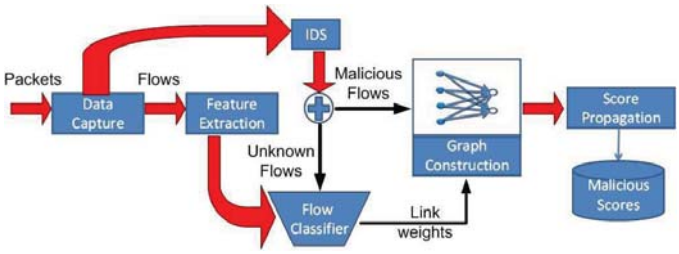

Fig. 2. Architecture of the proposed system, which uses the flow classification outputs from an IDS and a biased support vector machine classifier to construct a weighted directed graph for applying the score propagation algorithm used to compute the malicious scores of ISP clients.

IV. Methodology

This section presents the detailed methodology of our proposed system. First, we describe the construction of the HTTP communication graph from the network traffic flow data. We then discuss the flow classification and score propagation components of the system.

A. Graph Construction

Let $G = (\mathcal{V}, \mathcal{E})$ denote a directed graph constructed from the HTTP connections of our network trace, where $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ is the set of nodes associated with the IP addresses of the clients and servers and $\mathcal{E}$ is the set of directed links. Each link, $e_{ij} = (v_i, v_j)$, is an ordered pair of nodes, where the head of the link v_j corresponds to a server node and the tail of the link v_i corresponds to the client node. Note that the majority of the nodes derived from our network trace are either servers or clients with only a handful of them (less than 0.3%) have both incoming and outgoing links. Each node v_i is associated with a numeric score $y_i \in [0, 1]$ that represents its threat level or malicious score. Each link e_{ij} is associated with a set of flows $\pi_{ij} = \{f_{ij1}, f_{ij2}, \cdots, f_{ij|\pi_{ij}|}\}$, where $|\pi_{ij}|$ is the number of flows between the client-server node pair. Furthermore, if $\pi_{ij} = \emptyset$, then $(v_i, v_j) \notin \mathcal{E}$.

The link weight depends on the classification of its associated flows. First, we apply an IDS to determine whether the flows associated with the link are malicious. If any of the flows f_{ijk} is flagged as malicious, then the link weight is set to 1. Otherwise, a flow-based classifier is invoked. Let $\Sigma_{ij} = \{\sigma_1, \sigma_2, \cdots, \sigma_{|\pi_{ij}|}\}$ be the outputs of the flow-level classifier when applied to the flows in π_{ij}, where each $\sigma_l \in [0, 1]$. The link weight is determined as follows:

$$w_{ij} = \begin{cases} 1, & \text{if } \exists f_{ijk} \in \pi_{ij} : \mathcal{I}(f_{ijk}) = 1; \\ \max_{\sigma_k \in \Sigma_{ij}} \{\sigma_k\}, & \text{otherwise.} \end{cases} \quad (1)$$

where $\mathcal{I}(\cdot)$ denote the output of the IDS, which is equal to 1 if the flow is flagged as malicious, and zero otherwise. In other words, the weight of a link is given by the maximum score assigned to its flows according to either the IDS or the flow-based classifier. This approach is considerably different from previous graph-based methods applied to security-related

problems, in which the link weight either represents the presence or absence of a relationship or weighted according to the similarity between nodes [16][17]. The latter approach is inapplicable here since there is no node attribute available in our HTTP communication graph that would allow us to compute the pairwise similarity between the nodes. Instead, we only have a set of flows associated with each link in the network. Equation (1) enables us to determine the link weight based on the flow classification results. Furthermore, we consider the case when the link weight is binary-valued (0 or 1) as the baseline for comparison in our experiments. Our experimental results showed that the links weighted according to the flow-based classification outputs significantly outperformed the binary weight approach.

Finally, each node v_i in the HTTP communication graph is also assigned an initial threat level y_i^0 based on the IDS classification of its corresponding flows. If one of its flows is flagged as malicious by the IDS, then $y_i^0 = 1$. Otherwise, its threat level is initialized to zero. The initial scores of the nodes are used to instantiate the flow-based score propagation algorithm described in Section IV-C.

B. Classification of Network Flows

Our proposed system employs a flow-based classifier for two reasons. First, it could detect suspicious flows whose characteristics resemble those of known malwares but are missed by the IDS. Second, the classifier produces a continuous-valued output σ that reflects the likelihood of a flow to be malicious. The output can be normalized to have a range between 0 and 1 and used as a measure for the link weight, as shown in Equation (1).

The flow-based classifier is constructed using features extracted from the HTTP payload[3]. Constructing such a classifier is a non-trivial task due to the following two challenges:

- *Imbalanced class distribution*. The proportion of network traffic flows belonging to the unknown class far exceeds those belonging to the malicious class.
- *Positive labeled data only*. An IDS provides reliable labels for the positive class only. The class labels for the rest of the flows are unknown since they can be non-malicious or correspond to new malwares whose signatures are unavailable to the IDS.

To address these challenges, we employ a transductive biased support vector machine (SVM) classifier [19][20][21] to classify the network traffic flows. Transductive biased SVM allows the labeled data to contain only examples from a single (positive) class. Users can also specify a penalty function to vary the cost of misclassifying instances from different classes when training the classifier. In this work, the cost for misclassifying malicious instances as unknown outweigh the cost for misclassifying unknown instances as malicious.

Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ denote the training set used for building the classifier, where $\mathbf{x}_i$ is the feature vector for flow i and y_i is its corresponding class label. The transductive biased SVM classifier uses a linear model $h(\mathbf{x}) = \omega^T \phi(\mathbf{x}) + b$ to classify incoming flows, where ω and b are the model parameters. The classifier is trained to classify instances using a linear separating hyperplane in a high-dimensional feature space ϕ. The model parameters are estimated by solving the following constrained optimization problem:

$$
\begin{aligned}
\min_{\omega,b,\mathbf{y}^u} \quad & \frac{1}{2}\omega^T\omega + C^l \sum_{i=1}^{p} \xi_i^l + C^u \sum_{j=p+1}^{m} \xi_j^u \\
\text{s.t.} \quad & y_i^l(\omega^T \phi(x_i^l) + b) \geq 1 - \xi_i^l, \quad \forall i \in \{1, \cdots, p\} \\
& y_j^u(\omega^T \phi(x_j^u) + b) \geq 1 - \xi_j^u, \quad \forall j \in \{p+1, \cdots, m\} \\
& \xi_i^l \geq 0, \ \forall i \in \{1, \cdots, p\} \\
& \xi_j^u \geq 0, \ \forall j \in \{p+1, \cdots, m\}
\end{aligned} \tag{2}
$$

where m is the total number of flows collected and p is the number of flows labeled as malicious by the IDS. The superscripts l and u refer to the labeled (malicious) and unlabeled (unknown) examples. To build the model, the algorithm first trains on the labeled data only. It then applies the model to classify all the unlabeled instances. A threshold τ is then chosen to determine the high confidence unlabeled instances to be relabeled as positive class. This procedure is repeated until there are no significant changes in the labeling.

The transductive biased SVM classifier is flexible because it accommodates variable cost parameters to deal with the imbalanced class problem. Specifically, it uses the parameter C^l as the cost for misclassifying positive examples (i.e., malicious flows) and C^u as the cost for misclassifying unknown examples (i.e., non-malicious flows). By assigning $C^l > C^u$, this will help guide the classifier towards classifying more accurately training examples that belong to the positive class. The values of these parameters can be automatically chosen based on their class proportions or via cross-validation. The optimization problem can be solved in its dual form using quadratic programming. This requires the specification of a kernel function that measures the similarity between the training examples in a projected Hilbert space[22]. A popular choice for kernel function is the gaussian kernel, which is defined as $K(\mathbf{x}, \mathbf{x}') = \exp\left[-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\gamma^2}\right]$.

C. Flow-based Score Propagation

The score propagation algorithm identifies previously undetected malicious nodes in a network by taking into consideration the malicious scores of their neighbors (which in turn, depend on the scores of their respective neighbors, and so on). Similar to other graph-based methods [10] [11], it performs a random walk on a graph to iteratively propagate the scores across the network. The algorithm is based on the assumption

[3]In this work, we have extracted features from the payload of all unencrypted HTTP connections in the traces. We created 270 features per HTTP connection, which include *hostname length, hostname randomness [18], user agent, file name requested, file extension, content length, and server response code.* We consider only unencrypted HTTP traffic. If the traffic is encrypted, the features for the flow classifier will not be available. Nevertheless, there are other information such as layer-4 traffic information, DNS query information, *etc.*, that can be utilized to train the classifier. In principle, the flow classifier can be designed to use these features for classifying encrypted traffic flows.

that nearby nodes are likely to share similar class labels. While such an assumption still holds for our case, the scores must be propagated more judiciously, taking into account the threat levels of the flows associated with the links, to avoid generating high false alarm rates. This can be accomplished by using the flow classification outputs as the link weights for the HTTP communication graph. As noted in Section IV-A, if the flows associated with the link look suspicious, the weight of the link increases, thereby increasing the proportion of malicious score transmitted via the link.

More formally, our flow-based score propagation algorithm is designed to minimize the following objective function:

$$\begin{aligned} Q(\mathbf{y}) &= \frac{1}{2}\sum_{ij} W_{ij}\left[\frac{y_i}{\sqrt{D_{ii}}} - \frac{y_j}{\sqrt{D_{jj}}}\right]^2 \\ &+ \frac{\mu}{2}\sum_i (y_i - y_i^{(0)})^2, \end{aligned} \quad (3)$$

where $\mathbf{W}$ is a $|\mathcal{V}| \times |\mathcal{V}|$ matrix computed using Equation (1) and $\mathbf{D}$ is a diagonal matrix whose diagonal elements are given by $D_{ii} = \sum_j W_{ij}$. The column vector $\mathbf{y}$ represents the threat levels of the nodes and $\mathbf{y}^{(0)}$ is its initial value determined from the outputs of the IDS. Specifically, if a flow between client i to server j is detected to be malicious by an IDS, then both $y_i^{(0)}$ and $y_j^{(0)}$ will be initialized to 1. Otherwise, they are set to 0. Intuitively, the first term in the objective function ensures that the malicious scores for any pair of nodes connected by a highly weighted link should not differ substantially. The second term ensures that the scores of the nodes should not deviate significantly from their initial values. The objective function is similar to the one used in [16] for graph-based semi-supervised learning, except our weight matrix is computed from the flow classification outputs rather than the similarity between node attributes.

Next, we will illustrate the difference between the proposed score propagation algorithm and existing random walk algorithms such as PageRank. Our objective function can be rewritten in the following form:

$$\begin{aligned} Q(\mathbf{y}) &= \frac{1}{2}\left[\sum_i y_i^2 - 2\sum_{ij} y_i \frac{W_{ij}}{\sqrt{D_{ii}}\sqrt{D_{jj}}} y_j + \sum_j y_j^2\right] \\ &+ \frac{\mu}{2}\sum_i (y_i - y_i^{(0)})^2 \end{aligned} \quad (4)$$

This can be further simplified into matrix notation as:

$$\begin{aligned} Q(\mathbf{y}) &= \mathbf{y}^T(\mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})\mathbf{y} + \frac{\mu}{2}\|\mathbf{y} - \mathbf{y}^{(0)}\|^2 \\ &= \mathbf{y}^T\hat{\mathbf{L}}\mathbf{y} + \frac{\mu}{2}\|\mathbf{y} - \mathbf{y}^{(0)}\|^2 \end{aligned} \quad (5)$$

where $\hat{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}D^{-\frac{1}{2}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}}$ is the normalized graph Laplacian matrix of the network.

To optimize the objective function, we take its partial derivative with respective to y and set it to zero:

$$\begin{aligned} \frac{\partial Q(\mathbf{y})}{\partial \mathbf{y}} &= (\mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})\mathbf{y} + \mu(\mathbf{y} - \mathbf{y}^{(0)}) \\ &= (1+\mu)\mathbf{y} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}\mathbf{y} - \mu\mathbf{y}^{(0)} = 0 \end{aligned}$$

The preceding equation can be re-written in the form of an iterative update formula:

$$\mathbf{y} = \frac{1}{1+\mu}\hat{\mathbf{W}}\mathbf{y} + \frac{\mu}{1+\mu}\mathbf{y}^{(0)} = \beta\hat{\mathbf{W}}\mathbf{y} + (1-\beta)\mathbf{y}^{(0)} \quad (6)$$

where $\hat{\mathbf{W}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrized weighted adjacency matrix and $\beta = \frac{1}{1+\mu}$ controls the tradeoff between biasing the scores towards to the graph structure as opposed to the initial vector $\mathbf{y}^0$. If $\beta = 0$, the malicious scores are equal to the initial values obtained from IDS. On the other hand, if $\beta = 1$, the malicious score of a node depends only on the scores of its neighbors. For Web graphs, β is often set to 0.85 [12]. We use the same value in our experiments.

Though Equation (6) looks similar to the formula used in the PageRank algorithm [12], there is a subtle difference between the two in terms of how the adjacency matrix is normalized. In PageRank and other random-walk based algorithms, the adjacency matrix is transformed into a transition probability matrix with the following normalization $\hat{\mathbf{W}}_{rw} = \mathbf{D}^{-1}\mathbf{W}$, to ensure each row of the matrix sums up to 1. Instead, our approach normalizes the adjacency matrix by dividing each entry in the matrix with the degrees of both nodes connected by the link. As will be shown in our experiments (see Section VI-C), this normalization strategy helps to reduce the ill-effects of popular server nodes that have a large number of incoming connections, which can lead to high false alarm rates when applying the PageRank algorithm.

Finally, we consider a two-phase flow-based score propagation algorithm, which is more suited for the near-bipartite graph constructed for this domain. To simplify the discussion, we first assume the client and server nodes are disjoint. The symmetrized weighted adjacency matrix can be decomposed into the following block structure: $\hat{W} = \begin{bmatrix} 0 & \hat{W}^{(sc)} \\ \hat{W}^{(cs)} & 0 \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} \mathbf{y}^{(c)} \\ \mathbf{y}^{(s)} \end{bmatrix}$ and $\mathbf{y}^{(0)} = \begin{bmatrix} \mathbf{y}^{(c)(0)} \\ \mathbf{y}^{(s)(0)} \end{bmatrix}$, where $\hat{W}^{(sc)}$ is the adjacency matrix obtained by reversing the direction of the links from servers to clients. The update formula can be simplified to the following two subproblems:

$$\mathbf{y}^{(c)} = \beta\hat{\mathbf{W}}^{(sc)}\mathbf{y}^{(s)} + (1-\beta)\mathbf{y}^{(c)(0)} \quad (7)$$

$$\mathbf{y}^{(s)} = \beta\hat{\mathbf{W}}^{(cs)}\mathbf{y}^{(c)} + (1-\beta)\mathbf{y}^{(s)(0)} \quad (8)$$

where $\mathbf{y}^{(c)}$ and $\mathbf{y}^{(s)}$ are the score vectors for the two disjoint sets of nodes (e.g., clients and servers) in the bipartite graph. Our flow-based score propagation approach is now a two-phase algorithm. During the client propagation phase, the scores of the clients will be propagated to the servers by following their outgoing links. During the server propagation phase, the scores of the servers will be propagated to the clients in the reverse direction (i.e., by traversing in the opposite direction of their incoming links). Since most of the nodes are either clients or servers, their scores get updated only once in each round (during the client or server propagation phases). For nodes that are both clients and servers, their scores will be updated twice in each round to take into account the malicious scores of their neighboring servers and clients.

The pseudocode of our score propagation method is shown in Algorithm 1. The run-time complexity is $O(K(|\mathcal{V}|_s|\mathcal{V}|_c)$, where K is maximum number of iterations, $|\mathcal{V}_s|$ and $|\mathcal{V}_c|$ are the number of server and client nodes in the graph, respectively. For sparse matrices, the complexity can be reduced to $O(K|\mathcal{E}|)$, where $|\mathcal{E}|$ is the number of edges in the network.

Algorithm 1 Flow-based Score Propagation

Input: $G = (V, E)$, $\mathbf{y}^{(0)}$, and maximum iteration K
Output: $\mathbf{y}=\begin{bmatrix}\mathbf{y}^{(c)}\\ \mathbf{y}^{(s)}\end{bmatrix}$, malicious scores of the nodes in G
Construct $\hat{W}$ using Equation (1)
for $i = 1$ to K **do**
  Update client scores $\mathbf{y}^{(c)}$ using Equation (7)
  Update server scores $\mathbf{y}^{(s)}$ using Equation (8)
end for
Return $\mathbf{y}$

V. Experimental Setup

This section describes the data sets, our methodology for verifying the ground truth labels, and the evaluation metrics used to compare the performance of various approaches.

A. Data Sets

We evaluated the proposed system on a network trace collected at the edge of a point of presence (PoP) in a large Commercial ISP. There are close to 20000 distinct IP addresses inside the PoP. We focus our analysis on four 1-hour data sets from the original trace, which we denote as $D1$, $D2$, $D3$ and $D4$. We provide the detailed information in Table I.

The initial labels for the nodes are obtained using a commercial signature-based IDS. The IDS assigns a threat ID to those HTTP connections where the payload matches a predefined signature. In our traces, the IDS has identified HTTP connections matching traffic from various threats, such as *Skintrim, Sality, Conficker, Tidserv and Spyeye*. The number of malicious flows as well as malicious clients and servers detected using the IDS are shown in Table I.

B. Ground Truth Labels

Finding reliable ground truth for the nodes is a difficult problem by itself. In this work, we rely on our domain experts to confirm that the undetected malicious clients discovered by our approach are indeed malicious. The following data sources were used by the domain experts to verify the results: (i) Google's SafeBrowsing [23]. A service that Google provides to check for malware and phishing Web sites. One problem with SafeBrowsing is that it only stores sites that have been recently discovered as malicious (up to 90 days). For sites that were malicious more than 90 days in the past and are now offline or that have been cleaned up, SafeBrowsing may not provide any information; (ii) Web of Trust (WOT) [24]. A service that calculates the reputation of sites based on ratings from Web users and technical sources. WOT provides four different scores—trustworthiness, vendor reliability, privacy and child safety. Each of them ranges from 0 to 100, with 0 being most malicious. In our evaluation, we use only the first three scores from WOT ignoring the child safety score. A problem with WOT is that some sites, such as those used for rogue adwares, may be given a very low reputation score. Also, WOT scores can be missing for some sites. (iii) Blacklists. We use four blacklists from malwaredomainlist [25], malwaredomains [26], phishtank [27] and zeus-tracker [28]. In general, blacklists are known to have high false negatives.

C. Evaluation Metrics

Our analysis focuses on classifying previously undetected clients inside the network. To verify if a given client is suspicious, we first check if it connects to any known malicious Web sites. More specifically, if the IP address or the hostname of the Web site appears in SafeBrowsing or any blacklist, the site is flagged as malicious, and subsequently, any clients initiating connection to it is also considered suspicious. If none of the sites contacted by the client is malicious, we examine the sites' WOT scores. We consider a site to be malicious if its maximum WOT score[4] is below 10.

To increase our confidence in verifying whether a client is indeed malicious, we apply a minimum threshold (Num) on the number of the sites with low WOT scores contacted by the client. If the number of sites exceeds the threshold, the client is flagged as malicious. We report our experimental results for $Num = 1, 5, 10$. We rank the clients according to the malicious score after score propagation and compared them to the list of flagged clients. The precision of top n ranked clients is used to indicate the performance of different algorithms. We vary n from 1 to 200 and plot the precision curves.

VI. Experimental Results

In this section, we compared the performance of our proposed flow-based score propagation algorithm against two baseline approaches. The first baseline, termed as as *link-only* method, is used in Sections VI-A and VI-B to demonstrate the effectiveness of using flow-based classifier's output to weigh the links of the network. The link-only method is similar to our approach except it uses a binary 0/1 weight for its links. The second baseline, used in Section VI-C, considers the same normalization strategy as PageRank and other random walk methods. The link weights for the graph used by the second baseline method are provided by the flow-based classifier, which is equivalent to our score propagation method.

All the experiments were performed on a single PC with Intel Core i7 CPU 2.67 GHz and 8GB memory, running Windows 7 operating system. The run time on the datasets D1 $\sim$ D4 is shown in Table I. Even with the largest dataset, the algorithm converges in less than 190 seconds.

A. Performance Comparison for all Clients

Figure 3 compares the average precision of our flow-based score propagation approach against the link-only method. Each row corresponds to one of the four data sets used in this study while the columns show the results when the Num threshold

[4]WOT considers a Web site with any score below 20 as *very poor*.

TABLE I
DESCRIPTION OF FOUR HOURLY DATA SETS (D1 ∼ D4) USED IN OUR EXPERIMENTS

Data set	Number of Edges	Number of malicious flows/ Total number of flows	Number of malicious clients/ Total number of clients	Number of malicious servers/ Total number of servers	Run Time (seconds)
D1	377699	1088 / 1926620	26 / 5856	25 / 25627	142.8
D2	419502	1649 / 973271	30 / 6533	25 / 26346	167.4
D3	478449	1736 / 1033292	26 / 7360	22 / 26216	171.6
D4	525438	65 / 1078765	21 / 7936	23 / 27518	184.9

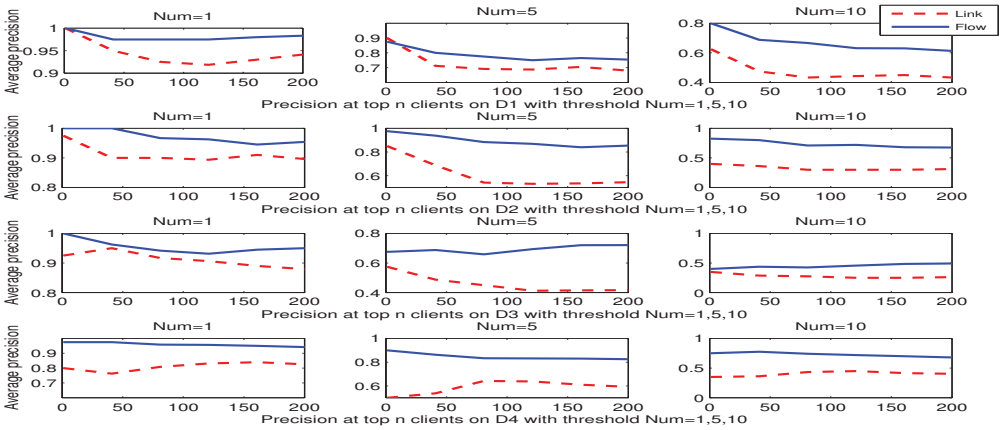

Fig. 3. Average precision at top n clients comparison between link-only and flow based methods on data $D1 \sim D4$

(described in Section V-C) is varied from 1 to 10. In each diagram, the horizontal axis corresponds n (the number of highest ranked clients examined) while the vertical axis shows the average precision of the top-n selected clients. The results suggest that both approaches can identify malicious clients that were previously undetected by the IDS, which had found only between 21-30 malicious clients on the different data sets (see column 4 in Table I). Furthermore, our flow-based approach significantly outperforms the link-only method on all four datasets. For example, out of the 5856 clients in $D1$, if we consider the precision for top 200 clients with $Num = 1$, our proposed method achieves a precision value close to 98% whereas the precision for the link-only method is around 94%. When we set the confidence threshold to $Num = 5$, the precision for our approach is close to 75%, whereas the precision for link-only method drops significantly to less than 70%. The difference is even more pronounced when we increase Num to 10. Note that a higher threshold for Num means a stricter condition for labeling a client as malicious. In all cases, our results clearly demonstrate the superiority of our flow-based score propagation method.

Figure 4 shows an example illustrating the advantage of using flow-based instead of link-only propagation. The plot considers only nodes located two hops away from the client node #102051. The red nodes in the diagram represent those were flagged as malicious by the IDS while the grey nodes represent the unlabeled ones. In this case, node #102051 was not classified as malicious by the IDS nor the link-only method. even though it was later verified to be malicious. The node is connected to a server node (#108194) that was flagged as malicious by the IDS. However, the rest neighbors of node #102051 were not flagged by the IDS. Since the weights of

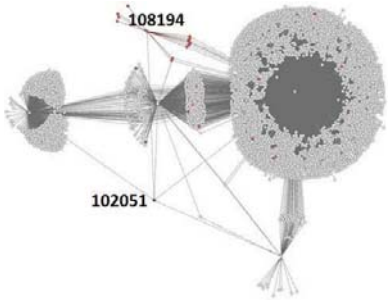

Fig. 4. HTTP communication subgraph around node #102051

the links are the same, the scores propagated to the client from other server nodes using link-only method are low, which is why the link-only method fails to identify it as malicious. On the other hand, the flow-based approach assigns a higher weight to the link between nodes #108194 and #102051, allowing a significant proportion of the malicious scores to be transmitted along this link. This raises the malicious score of node #102051 significantly, making it one of the highest ranked nodes after flow-based propagation. Subsequent investigation confirmed that client #102051 is indeed infected.

B. Performance Comparison of Clients Indirectly Connected to Malicious Sites

The results shown in the previous section considers all the clients, including those that are directly connected to nodes flagged as malicious by the IDS. In this subsection, we analyze the results for identifying previously undetected malicious clients that are not directly connected to any malicious nodes labeled by the IDS. For example, in data set $D2$, there are 176 such clients identified among the top 200 clients found by the link-only method, while in the ranking provided by

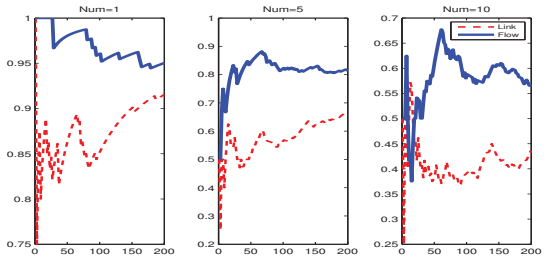

Fig. 5. Average precision at top n highest ranked clients that are indirectly connected to malicious hosts

the flow based method, there are 187 such clients. The top-200 precision for both methods are reported in Figure 5, which looks quite similar to the results shown in the previous subsection. The precision is relatively high in top 200 clients, which suggests that the algorithm places most of the malicious clients at the top of the ordering. The result also confirms that our proposed method is not only capable of finding malicious clients directly connected to nodes flagged by the IDS but also those located further away.

C. Comparing Flow-based Score Propagation against Random Walk Propagation

As mentioned in Section IV-C, the proposed flow-based score propagation differs from PageRank and other random walk algorithms in terms of how the adjacency matrix is normalized. Our proposed normalization is desirable because it takes into consideration the in-degree of the server nodes. Thus, popular servers with high in-degrees will have their link weight significantly reduced compared to random walk based methods. For example, a popular website like $google.com$, will be highly ranked using the random walk based normalization due to its high in-degree. However, the symmetric normalization used by our method was able to reduce its malicious score considerably. Finally, among the top 100 highest ranked server nodes detected by the flow-based score propagation method, the precision for symmetric normalization is 63%, while the precision using PageRank type of normalization is only 51%.

VII. Conclusions

This paper presents a novel approach to combine the HTTP communication graph with flow information to effectively identify malicious ISP clients not identified by the IDS. Our approach employed a flow-based score propagation algorithm to identify previously undetected malicious clients in the monitored network. Experimental results using real-world data from a large Commercial ISP showed that the proposed method helps to enhance the detection of malicious clients without incurring a high false alarm rate due to popular servers.

Although the results look promising, the methodology can still be improved in several ways. First, the flow classifier can be extended to incorporate layer-4 features in order to deal with encrypted HTTP traffic. Second, although it can effectively detect malicious clients, we can extend it to detect malicious servers that reside outside the monitored network and only have partial HTTP connectivity profile available.

References

[1] "Symantec internet security threat report," Symantec Corporation, http://www.symantec.com/threatreport/, Tech. Rep. Volume 18, 2013.
[2] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious urls," in *Proc. of the ACM SIGKDD*, 2009, pp. 1245–1254.
[3] M. Akiyama, T. Yagi, and M. Itoh, "Searching structural neighborhood of malicious urls to improve blacklisting," in *Proc. of 11th Annual Int'l Symposium on Applications and Internet*, 2011, pp. 1–10.
[4] J. W. Stokes, R. Andersen, C. Seifert, and K. Chellapilla, "Webcop: locating neighborhoods of malware on the web," in *Proc. of Int'l Conference on World Wide Web*, 2011, pp. 187–196.
[5] A. Le, A. Markopoulou, and M. Faloutsos, "Phishdef: Url names say it all," in *Proc. of the 30th IEEE Int'l Conference on Computer Communications*, 2011, pp. 191–195.
[6] J. Zhang, C. Seifert, J. W. Stokes, and W. Lee, "Arrow: Generating signatures to detect drive-by downloads," in *Proc. of the 20th Int'l Conference on World Wide Web*, 2011, pp. 187–196.
[7] G. Jacob, R. Hund, C. Kruegel, and T. Holz, "Jackstraws: picking command and control connections from bot traffic," in *Proc. of the 20th USENIX Security Symposium*, 2011.
[8] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "Bothunter: detecting malware infection through ids-driven dialog correlation," in *Proc. of the 16th USENIX Security Symposium on USENIX Security Symposium*, 2007, pp. 12:1–12:16.
[9] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proc. of USENIX Security Symposium*, 2008, pp. 139–154.
[10] V. K. Stanford and V. Krishnan, "Web spam detection with anti-trust rank," in *Proc. of the Second Int'l Workshop on Adversarial Information Retrieval on the Web*, 2006, pp. 37–40.
[11] J. Zhang, P. Poras, and J. Ullrich, "Highly predictive blacklisting," in *Proc. of USENIX Security Symposium*, 2008, pp. 107–122.
[12] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks*, vol. 30, no. 1-7, pp. 107–117, 1998.
[13] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting botnet command and control channels in network traffic," in *Proc. of the Network and Distributed System Security Symposium*, 2008.
[14] P. M. Comar, L. Liu, S. Saha, A. Nucci, and P.-N. Tan, "Weighted linear kernel with tree transformed features for malware detection," in *Proc. of the 21st ACM International Conference on Information and Knowledge Management*, 2012, pp. 2287–2290.
[15] P. M. Comar, L. Liu, S. Saha, P.-N. Tan, and A. Nucci, "Combining supervised and unsupervised learning for zero-day malware detection," in *Proc. of the 32nd IEEE International Conference on Computer Communications*, 2013, pp. 2022–2030.
[16] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Scholkopf, "Learning with local and global consistency," in *Proc. of Advances in Neural Information Processing Systems*, 2004, pp. 321–328.
[17] L. Liu and P.-N. Tan, "A framework for co-classification of articles and users in wikipedia," in *Proc. of International Joint Conferences on Web Intelligence and Intelligent Agent Technologies*, 2010.
[18] S. Yadav, A. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *Proc. of the 10th ACM Conference on Internet Measurement*, 2010, pp. 48–61.
[19] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. of the Sixteenth Int'l Conference on Machine Learning*, 1999, pp. 200–209.
[20] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu, "Building text classifiers using positive and unlabeled examples," in *Proc. of the 3rd IEEE Int'l Conference on Data Mining*, 2003.
[21] L. Liu, P. M. Comar, S. Saha, P.-N. Tan, and A. Nucci, "Recursive nmf: Efficient label tree learning for large multi-class problems," in *Proc. of the 21st International Conference on Pattern Recognition*, 2012.
[22] L. Liu, P. M. Comar, A. Nucci, S. Saha, and P.-N. Tan, "Missing or inapplicable: Treatment of incomplete continuous-valued features in supervised learning," in *Proc. of SDM*, 2013, pp. 46–54.
[23] "GoogleSafeBrowsing," https://developers.google.com/safe-browsing/.
[24] "WOT," http://www.mywot.com/.
[25] "MalwareDomainList," http://www.malwaredomainlist.com/.
[26] "Malwaredomains," http://www.malwaredomains.com/.
[27] "Phishtank," www.phishtank.com.
[28] "Zeus-tracker," https://zeustracker.abuse.ch/.