Content-Agnostic Malware Detection in Heterogeneous Malicious Distribution Graph *

Ibrahim Alabdulmohsin
King Abdullah University of Science & Technology
ibrahim.alabdulmohsin@kaust.edu.sa

Yufei HAN
Symantec Research Labs
yufei_han@symantec.com

Yun Shen
Symantec Research Labs
yun_shen@symantec.com

Xiangliang Zhang
King Abdullah University of Science & Technology
xiangliang.zhang@kaust.edu.sa

ABSTRACT

Malware detection has been widely studied by analysing either file dropping relationships or characteristics of the file distribution network. This paper, for the first time, studies a global heterogeneous malware delivery graph fusing file dropping relationship and the topology of the file distribution network. The integration offers a unique ability of structuring the end-to-end distribution relationship. However, it brings large heterogeneous graphs to analysis. In our study, an average daily generated graph has more than 4 million edges and 2.7 million nodes that differ in type, such as IPs, URLs, and files. We propose a novel Bayesian label propagation model to unify the multi-source information, including content-agnostic features of different node types and topological information of the heterogeneous network. Our approach does not need to examine the source codes nor inspect the dynamic behaviours of a binary. Instead, it estimates the maliciousness of a given file through a semi-supervised label propagation procedure, which has a linear time complexity w.r.t. the number of nodes and edges. The evaluation on 567 million real-world download events validates that our proposed approach efficiently detects malware with a high accuracy.

1. INTRODUCTION

Modern cyber criminals employ various sophisticated mechanisms to drop malicious files such as adware, ransomware, and rouge antivirus. They can set up short lived websites (e.g., using domain fluxing), hire a third party (e.g., payper-install), or utilise exploit kits (e.g., drive-by download) to deliver the payloads. In recent years, malware delivered to the endpoints uses various antivirus evasion techniques to avoid detection. These include environment awareness, timing-based evasion, repackaging internal data obfuscation and so on.

Despite all the previous efforts [5, 18, 10, 4, 14, 9, 15, 6, 3, 7, 8], how to quickly identify if a new binary is malicious or benign remains a crucial problem yet to be satisfactorily solved. Naturally, binary analysis techniques are effective solutions to this problem as static [15] and dynamic analysis [3, 8] can offer deep insights of a binary via thorough low-level examinations. However, such techniques need considerable amount of computational resources to carry out the inspection. The sheer volume of suspicious binaries, together with the repackaging and obfuscation techniques used by cyber criminals make the task of timely analysing binary files nearly impossible. Even though recent research efforts in large-scale malware analysis (e.g., [6], [7]) are proven effective in uncovering clusters of malware that share known low-level similarity, they are less effective and computationally costly in identifying unknown malware or threats.

On the other hand, content-agnostic approaches have advantages in coping with the large and heterogeneous space of malicious binaries by avoiding content inspection, hence becoming more popular in recent years to compliment binary analysis approaches. Previous work in this direction detects malware from various perspectives, e.g., understanding the characteristics of malware delivery infrastructures [2, 14], identifying topological relations among hosts and IPs in the malicious Web infrastructure used by cyber criminals [18, 10, 5], proactively and efficiently crawling potential malicious hosting sites [4], mining file co-occurrence relationship [16], getting insights from downloader-payload relationship of executable files on an endpoint [9], and so on.

It is interesting to note that the aforementioned work either focuses on analysing network characteristics of malware distribution [2, 14, 18, 10, 5] or centres the analysis on file dropping relationships from the endpoints [9]. It makes more sense to fuse the aforementioned file dropping graph and file distribution network to form a global view of end-to-end distribution network instead of evaluating whether a given file is malicious or benign from either perspective. This global view offers the integral ability of structuring the relationship in a scalable manner and handling heterogeneous sources of information. However, it brings a couple of intriguing challenges. It requires a method capable of working with the heterogeneous graph and, at the same time, the method should be computationally efficient to handle large scale graphs while offering a high accuracy.

The contributions of this paper are summarised as follows.

*All authors contributed equally and are ordered alphabetically.

CIKM'16 , October 24 - 28, 2016, Indianapolis, IN, USA

ISBN 978-1-4503-4073-1/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2983323.2983700

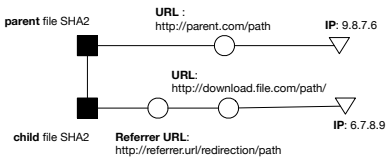


Figure 1: Abstract model of malware distribution graph.

- We provide a global heterogeneous malware delivery graph fusing file dropping relationship and the topology of the file distribution network.
- We unify topological information of heterogeneous network and content-agnostic features of different node types using a novel Bayesian label propagation model.
- We demonstrate our method is efficient and effective in identifying malicious files from over 567 million real-world download events.

Comparing to the state-of-the-art research work, our proposed approach has the following advantages:

- It only requires a small amount of labelled data to achieve high detection precision, thanks to its novel approach of combining topological information with content-agnostic features in a unified manner using Bayesian label propagation.
- The proposed approach is computationally efficient. It doesn't need to examine the source codes, nor inspect the dynamic behaviours of a binary. It estimates the maliciousness of a given file through a semi-supervised label propagation procedure. The proposed method's time complexity is $O((|E|+|V|)\times T)$, where $|V|$ is the number of nodes in the graph, $|E|$ is the number of edges in the graph, and T is the number of iterations.

2. METHODOLOGY

As mentioned earlier, the primary objective of this work is to be able to detect malware reliably in a content-agnostic manner. This has the advantage of yielding a computationally efficient solution, since signature-based low-level inspection of binary files is avoided, as well as providing a more robust solution to obfuscation techniques, such as polymorphism and metamorphism. In order to accomplish the desired goal, we propose to utilise both topological information, such as the dropping relationship of files, as well as content-agnostic features, such as lexical tokens in URLs and file names. Clearly, topological information and content-agnostic features are quite different in nature. Yet, we will show that they can be seamlessly combined in a *unified* manner using Bayesian label propagation. In brief terms, supervised classification via the content-agnostic features will form the *prior* to the Bayesian label propagation algorithm. Later, we will demonstrate experimentally that such a unified approach indeed outperforms label propagation and supervised classification when they are implemented separately.

2.1 Preliminary

To describe the malware detection problems in formal terms, suppose we have $\{(x_1, z_1, y_1)...(x_l, z_l, y_l)\}$ instance-type-label tuples, which are assumed to be drawn i.i.d. from some fixed unknown probability distribution $\mathcal{D}$ over the product space $\mathcal{X}\times\mathcal{Z}\times\{+1,-1\}$. Here, a positive label $y_i=+1$ implies that the instance $x_i\in\mathcal{X}$ whose type is z_i is malicious. The instance types are the types of nodes available in the malware distribution network (MDN), which, in our case, includes files, URLs, and IP addresses. In addition to the labeled data, we also have *unlabelled* instances $\{(x_{l+1}, z_{l+1}), \ldots, (x_{l+u}, z_{l+u})\}$, whose underlying (unknown) true class labels are denoted $\{y_{l+1}, ..., y_{l+u}\}$. The goal is to use the labeled data $\{(x_1, z_1, y_1), ..., (x_l, z_l, y_l)\}$ and the unlabelled instances $\{(x_{l+1}, z_{l+1}), \ldots, (x_{l+u}, z_{l+u})\}$ to predict the unknown labels $\{y_{l+1}, ..., y_{l+u}\}$. In other words, we operate in the *transductive* learning setting, as opposed to the *inductive* learning setting. Operating in the transductive learning setting is more appropriate for malware detection since malware distribution networks change over time, with new URLs, IPs, and files being created on a regular basis.

In our implementation, we classify node types into the following three categories (see Section 2.2 for a more detailed descriptions of the features used for each node type):

1. *Files*: Some of its features include the file size and average reputation scores of the workstations that downloaded the file.
2. *URLs*: Its features are mostly lexical in nature, which has been demonstrated in some recent work to be quite effective for URL classification [11]. Such lexical features include the URL length, the number of tokens in the fully-qualified domain name (FQDN), the number of common phishing words, such as `ebayisapi` and `webscr`, as well as a the presence or absence of explicit port numbers in the URL.
3. *IP Addresses*: Some of its features include the number of unique top-level domain names hosted by the IP address and the total number of files downloaded from the given IP address.

These three node types are clearly related to each other. Consequently, in addition to the content-agnostic features, we also assume that the relationships between nodes are captured in a graph. Hence, we have content-agnostic features as well as graph-based topological information that both aid the malware detection task.

The proposed approach is to first implement supervised classification for each node type separately. These classifiers output probabilistic scores, which, in turn, are used as priors to the label propagation algorithm carried out over the graph. We describe each stage of the process in details next.

2.2 Supervised Classification

The first stage of the proposed algorithm is to implement a binary classifier for each node type in the graph using its content-agnostic features. The goal of the classifier is to output a probability score if the given node is malicious or benign, i.e. $P(y_i=+1|x_i, z_i)$. These probability scores can be obtained, for example, using logistic regression, the naïve Bayes classifier, or support vector machine (SVM) with the Platt scaling [13]. The classification algorithm used in our

Table 1: Features for file, URL and IP classification.

Category	Features	Feature Description
Features for file classification	in/out degree	The number of landing pages to the file and files dropped by the file
	src/trgt reputation	The average reputation score of droppers/payloads (if any)
	src/trgt prevalence	The average prevalence score of droppers/payloas
	size	The size of the file in kilobytes (KB)
Features for URL classification	url len	The number of characters in the URL
	cnt. of dom and path token	The number of tokens in FQDN and in the path
	avg. of dom and path token	The average token length in the domain and in the path
	domain is ip	A binary indicator if an IP address is used directly, instead of a FQDN
	has port	A binary indicator if a port is present in the URL
	len. dom. token	The length of longest token in FQDN
	num of dots and hyph	The number of '.' and '-' in the URL
	has phish	The number of phishing words in the list `'webscr', 'secure', 'banking', 'ebayisapi', 'account', 'confirm', 'login', 'signing'`
	brand present	The name of a Fortune 500 company is present in the URL
	num of subd.	The number of subdomains given in the URL
Features for IP classification	size	The size of the subgraph rooted on the IP address
	num urls / rurls	The number of URLs/referrer URLs in the subgraph rooted on the IP address
	num files	The number of files in the subgraph
	uniq url tlds/subd	The number of unique top level domains/subdomains hosted by the IP address
	avg url/rurl hfiles	The average number of files hosted per URL / referred by a referrer URL
	rurl file cover	The total number of files referred by referrer URLs in the subgraph
	uniq rurl tlds / uniq rurl subd	The total number of files directly hosted in the URLs / unique subdomains
	avg/max/min url priori	The average/max/min score of URL priori in the subgraph
	avg/max/min file priori	The average/max/min score of file priori in the subgraph

implementation is 200-tree random forest for file, URL and IP, which is implemented using scikit-learn 0.17.1. The full list of features used is shown in Table 1.

2.3 Bayesian Label Propagation

Having obtained a binary classifier for each node type, the next stage is to use the predicted score as a prior to the label propagation algorithm. The basic idea of label propagation is to model the relations between instances using an undirected unweighted graph $G = (\mathcal{V}, \mathcal{E})$ and to use the adjacency matrix to infer the labels [1, 19, 17]. Each node $v_i \in \mathcal{V}$ corresponds to a single training instance-type pair (x_i, z_i). Each edge $e_{i,j} \in \mathcal{E}$ represents the relation between node v_i and v_j. As mentioned earlier, we have three node types in our implementation: files, URLs, and IP addresses, where the set of edges $\mathcal{E}$ indicate the file delivery relation. For example, if one file A is downloaded from URL B and the URL is hosted by the IP address C, the corresponding nodes should be linked in the graph G to encode this delivery relation. These relations are illustrated in Figure 1.

To derive a Bayesian version of label propagation, we closely follow the proof steps of the SocNL algorithm proposed in [17]. We will show how the supervised classifier can be incorporated as well in the form of a prior.

First, suppose that for every node in the graph $v_i \in \mathcal{V}$, there exists a Bernoulli parameter θ_i such that $P(y_i = +1) = \theta_i$. Also, suppose that the neighbours N_i of node v_i are labeled. The goal of label propagation is to compute $f_i = P(y_i = +1 \mid N_i)$. In line with the Bayesian spirit, we will achieve this by first computing:

$$P(\theta_i \mid N_i) = \frac{P(\theta_i) \cdot P(N_i \mid \theta_i)}{P(N_i)} \tag{1}$$

Next, as is used in the derivation of the SocNL algorithm [17], we impose a *smoothness* constrain on the graph. In particular, we assume that $\theta_i \approx \theta_j$ for all $v_j \in N_i$. With this approximation, we obtain:

$$P(N_i \mid \theta_i) = \prod_{v_j \in N_i} \theta_i^{\delta(y_j=+1)} \cdot (1-\theta_i)^{\delta(y_j=-1)} \tag{2}$$

Next, we use the beta distribution as a prior $P(\theta_i)$ since it is conjugate to the Bernoulli distribution. So, we write:

$$P(\theta_i) \propto \theta_i^{\alpha_i - 1} \cdot (1-\theta_i)^{\beta_i - 1}, \tag{3}$$

with shape parameters $\alpha_i, \beta_i > 0$. As a result, we obtain:

$$P(\theta_i \mid N_i) \propto \theta_i^{|N_i^+| + \alpha_i - 1} \cdot (1-\theta_i)^{|N_i^-| + \beta_i - 1}, \tag{4}$$

where $|N_i^+|$ and $|N_i^-|$ is the number of positive and negative neighbors respectively to node v_i. Finally, as shown in [17], we predict using marginalisation:

$$\begin{aligned} f_i &= P(y_i = +1 | N_i, \alpha_i, \beta_i) \\ &= \int_{\theta_i} P(y_i = +1 | \theta_i) \cdot P(\theta_i | N_i, \alpha_i, \beta_i)\, d\theta_i \\ &= \frac{|N_i^+| + \alpha_i}{|N_i| + \alpha_i + \beta_i} \end{aligned} \tag{5}$$

Intuitively, the prediction rule in Eq. (5) states that the probability $P(y_i = +1 | N_i)$ is estimated by counting the fraction of positively-labeled neighbours to node v_i, where the prior parameters α_i, β_i play the role of a bias.

Our next goal is to let the classifier determine the values of the hyper-parameters α_i and β_i. The output of the classifier, denoted $\hat{y}_i$, can be interpreted as the *mean* of θ_i. Since the marginal distribution of θ_i is a beta distribution, whose mean is $\alpha_i/(\alpha_i + \beta_i)$, we write $\gamma = \alpha_i + \beta_i$ and $\alpha_i = \gamma\, \hat{y}_i$. Note that $E[\theta_i] = \hat{y}_i$, which is the probability score predicted

by the classifier. Therefore, the prediction rule reduces to:

$$f_i = P(y_i = +1|N_i, \alpha_i, \beta_i) = \frac{|N_i^+| + \gamma\hat{y}_i}{|N_i| + \gamma} \quad (6)$$

Of course, this rule assumes that all of the neighbours of node v_i are labeled. However, it can be shown (see for instance Section 4.2 in [17]) that the rule above can be translated into an iterative label propagation rule of the form:

$$f_i = \frac{\sum_{v_j \in N_i} f_j \ + \ \gamma\hat{y}_i}{\sum_{v_j \in N_i} f_j \ + \ \gamma} \quad (7)$$

This is the label propagation algorithm used in our implementation. The LP update rules in Eq. (7) are guaranteed to converge as long as $\gamma > 0$ [17].

As we can see, the estimated label of an unlabelled instance is decided by the weighted average of label probability of its neighbours in addition to the probability estimate predicted by the supervised classifier. In this way, information about class membership is recursively propagated from labelled instances to unlabelled ones through the connected paths inside the graph G until convergence is achieved. Label propagation has advantages in terms of time complexity. The propagation procedure is conducted by simple and fast local linear weighted averaging operations, as shown in Eq. (7), which runs in $O((|E| + |V|) \times T)$ time, where $|V|$ is the number of nodes in the graph, $|E|$ is the number of edges in the graph, and T is the number of iterations.

3. DATASETS

3.1 Study Dataset and Ground Truth

Study Dataset. We use the user *download activity data* provided by Symantec. This dataset is collected from users who opt in for data sharing program, and client identifiers are anonymised. It is not possible to link back the collected data to the client from which the requests originated. The download activity data provide meta-information about all download activities on the endpoints. The dataset is further enriched using *binary reputation* to include metadata about reputation band and prevalence of the downloaded files. We extract over 567 million download events from 17th March to 11th April 2016 to form our study dataset.

Additional Datasets. We collect benign URLs and malicious URLs from various sources to build a training dataset for the URL node classifier (see Section 2). For benign URLs, we use the top 10,000 of 2015 most popular search words and get the top 6 URLs of each word from Google search results. We also get a list of benign URLs (whitelisted) from the security vendor's internal database. In total we have 87,974 benign URLs. For malicious URLs, we obtain these URLs from publicly available sources - cybercrime tracker, phishtank, and malc0de[1]. We also get a list of known malicious URLs from the security vendor's internal database. In total we have 36,036 malicious URLs.

Ground Truth. We query VirusTotal (a free online service that aggregates files and URLs scanning outputs from different antivirus engines) for each file SHA2 to obtain its *first seen* timestamp, the number of AV products (and associated vendor names) that flagged the file as malicious, and the total number of AV products that scanned the file. We consider a file malicious if *at least one* of the top five AV vendors (w.r.t. market share) and *a minimum of two other* AV vendors detect it as malicious [12]. We also obtained additional ground truth about files from the same major security company. The verdicts come from the company's static and dynamic binary analysis platform. In total, our ground truth consists of 4,868,770 benign and 1,544,215 malicious files. We summarise our datasets in Table 2.

[1]Respectively at http://cybercrime-tracker.net, https://www.phishtank.com, http://malc0de.com/database

Table 2: Graph data size (on average between 17th March and 11th April 2016), and URLs for priori classifiers training

Graph (Avg. per day)	No. of nodes			No. of edges
	file	URL	IP	
	1,648,458	892,394	155,533	4,047,741
Ground Truth files (Total)	4,868,770 benign, 1,544,215 malicious			
URL	87,974 benign, 36,036 malicious (external)			

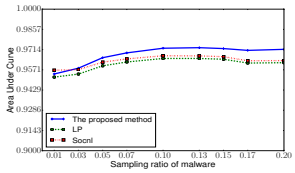
4. EXPERIMENTAL ANALYSIS

4.1 Experimental setup

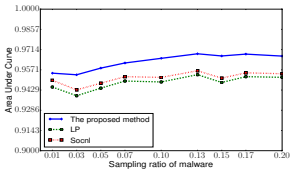
Our experiments are organised in two parts. In the first part, we aim to verify the benefits of the proposed method and compare its performance against both standard label propagation [19] and the Bayesian inference based variation of LP, named SocNL [17]. Note that we neither compare to NAZCA, due to its scalability issue in large graphs, nor other malware detection techniques as they have only partial information regarding malware distribution network. We traverse different sampling ratios of benign files and malicious files to identify potential malware with different numbers of initially labelled file nodes in the graph. Both Area Under ROC Curve (AUC) and F1-score of all three algorithms are used to evaluate detection performances quantitatively. The second part demonstrates how our method can identify malware with a real-world case study.

The datasets used to conduct the experimental study were summarised in Section 3. To conduct malware detection on day t, we use the features of benign files and malicious files that have been identified before day $t-1$ as the training set to build the file node classifiers. URL node specific classifier is trained independently using additional exteriorly labelled malicious and benign URL samples (see Section 3). To construct the IP-specific node classifier, we borrow the idea from self-training technology [1] and build the initial training set using March 16th data. That is, we run the proposed method on the March 16th graph with non-informative priors for IP nodes. Once the inference finishes, we set up a threshold on the derived probabilistic labels of the IP nodes. IP nodes with the probabilistic score larger/smaller than the threshold are recognised as a potential malicious/benign IP. We then extract features from these IP nodes and train the node classifier accordingly.

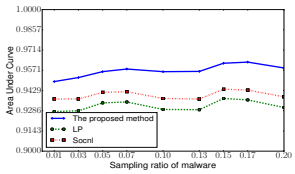
To choose the optimal setting of the regularisation parameter γ in Eq. (7), we carry out a grid search in the set {0.1, 0.01, 0.001, 0.0001} using cross validation, and report the results with the best detection performance.



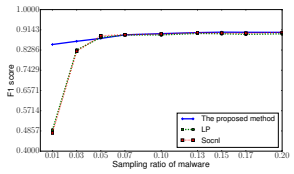
(a) Benign sampling rate = 0.3



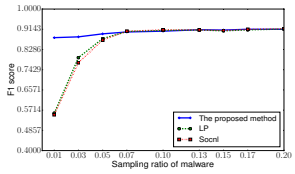
(b) Benign sampling rate = 0.5



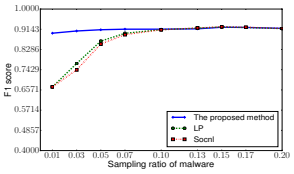
(c) Benign sampling rate = 0.7



(d) Benign sampling rate = 0.3



(e) Benign sampling rate = 0.5



(f) Benign sampling rate = 0.7

Figure 2: Average AUC and F1 scores with different number of initially labelled nodes.

Table 3: Comparison of F1 scores between file classifier and the proposed method

Date	File classifier	The proposed method
March 30	0.833	0.910
March 31	0.834	0.896
April 1	0.826	0.902
April 2	0.838	0.884

4.2 Comparative Study

To investigate how the proposed method performs given different numbers of initially labelled file nodes, we vary the benign file sampling ratio as 0.3, 0.5 and 0.7. For each level of benign file sampling ratio, we further choose 9 different malware sampling ratios, respectively 0.01, 0.03, 0.05, 0.07, 0.09, 0.11, 0.13, 0.15, 0.17, and 0.19. The ranges of the two sampling ratios are selected to reflect the real world scenario; there are plenty of benign files available while the number of malware samples is limited.

We use the settings of sampling initially labelled nodes for all three algorithms involved in the comparison and run all the algorithms three times on each day. Given a specific benign file sampling ratio and a fixed malware sampling ratio, AUC values and F1 scores derived from all 26 days are then averaged. The mean AUC and F1 score of each algorithm evaluate its overall malware detection performance under the given level of the sampling rates. Figure 2a, Figure 2b and Figure 2c show the mean AUC value derived by fixing the benign sampling rate and increasing gradually the malware sampling rate. Figure 2d, Figure 2e and Figure 2f shows the mean F1 scores following the same setting.

As shown in the figures, the proposed method provides *superior* detection accuracy over the other two algorithms when the number of sampled malware is limited, e.g., malware sampling rate less than 0.07, no matter how large the benign file sampling rate is. In terms of average AUC and F1 scores, the proposed method yields a more distinctive improvement of detection accuracy with increasingly larger benign file sampling rate. This observation is consistent with the target of the algorithmic design. Given limited number of initially labelled malware samples and imbalance between benign and malware samples, the node classifiers built on the heterogeneous MDN graph provide a discriminative prior over the class label of graph nodes. This node prior estimate provides a complementary supervised information to the label propagation procedure, which reduces the impact of the imbalance issue and lack of labelled malware samples. Such a property of the proposed method makes it well suited for solving challenging malware detection problems.

We measure also the average F1 score of the content-agnostic file-specific classifier with each level of malware sampling rate. Across different malware and benign file sampling rates, the average F1 score of the file-specific classifier is less than 0.83, distinctively lower than all the label propagation based methods in the comparison. In Table 3, we illustrate an example by comparing the F1 scores of the file specific classifier with those of the proposed method from March 30th to April 2nd, given benign file sampling rate as 0.5 and malware sampling rate as 0.05. These results indicate that the file specific classifier is unable to achieve a satisfying detection accuracy by itself. However, the probabilistic output of the classifier does produce complementary prior information to improve the detection accuracy of label propagation. Increasing the sampling rate of benign file over 0.5 results in a slight decrease in the AUC scores for all three algorithms. It is caused by the imbalanced issue of the training data, which increases slightly the type II error.

We run daily malware detection on the MDN graph, in order to simulate real-world malware filtering applications. The MDN graph generated per day has on average 1,648,458 file nodes, 892,394 URL node and 155,533 IP nodes, as summarised in Table 2. Average running time for malware detection on the large-scale MDN graph is 11 minutes, with 8 minutes for building the priori classifiers and 3 minutes for conducting label propagation concatenated with the priori information on the graph. Furthermore, as we observed in our experimental study, the running time of the whole malware detection increases linearly with the graph size. In practice, fast computing of the proposed method makes it possible to conduct semi-real time, e.g. hourly, malware detection on large-scale file collections.

4.3 Real-World Case Study

In this section, we provide a detailed case study showing how our proposed method is able to identify malicious files in a real-world malware distribution network excerpted from our March 17th graph. For the sake of simplicity, we use a tuple $(1 - f_i, f_i)$ in the figures to denote the label inference score of each node, where $f_i = P(y_i = +1 \mid N_i)$ and $f_i \in [0, 1]$ (see Section 2.3 for details). From our training data, node 1 is identified as information stealer malware 'limitail' $(0, 1)$ and node 3 is a legitimate windows 'calculator' application $(1, 0)$. Note that both files are hosted in the same site. This is a common strategy taken by cybercriminals to target different platforms; depending on various criteria set by these cybercriminals, either a benign file or malicious binary could be dropped. However, such strategy brings issues to graph-based inference methods. Take SocNL

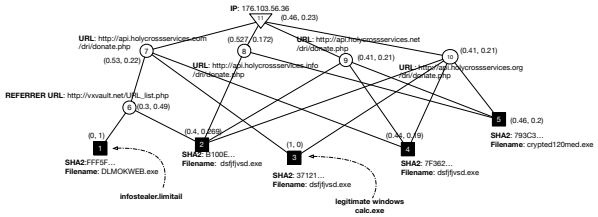


Figure 3: Real-world malware delivery network example: SocNL inference result.

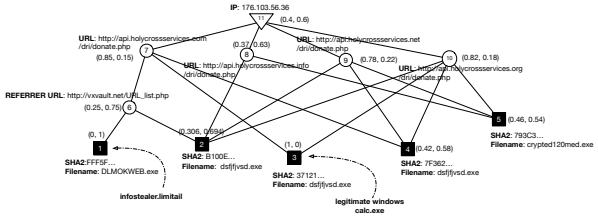


Figure 4: Real-world malware delivery network example: inference result by the proposed method.

for example, without a node priori information, SocNL can only use node 1 and node 3's initial labels leaving the rest of the nodes unlabelled. As we can see in Figure 3, SocNL is inclined to infer node 2, 4, and 5 as benign with high confidence scores. Thanks to the node classifiers (see Section 2 for details), our method is able to give each node *a priori* bias before label propagation begins. As we can observe in Figure 4, the proposed method is able to convict node 2, 4, and 5 as malicious. All three nodes are later confirmed in VirusTotal as 'Dridex' malware. Additionally, the proposed method is also able to convict node 8 as malicious. However, this graph connects to the whole malware distribution graph via a single node (node 1). This creates a propagation bottleneck and the proposed method is thus not able to convict the rest of the nodes.

5. CONCLUSION

In this paper, we propose a content-agnostic approach to detect malware by constructing a heterogeneous file distribution graph and leveraging semi-supervised learning on the built graph. Our work avoids the time-consuming binary analysis of malicious binaries and, at the same time, has the advantages of detecting malicious files accurately. It integrates exterior node classifiers for different node types (e.g., files, URLs and IP addresses) to reduce uncertainty during the inference process. The proposed approach is computationally scalable and can be easily parallelised for large-scale malware detection applications. We intend to measure the performance of the proposed method in dynamic networks where temporal factor is considered.

6. REFERENCES

[1] Y. Bengio, O. Delalleau, and N. L. Roux. *Label Propagation and Quadratic Criterion*, pages 193–216. MIT Press, 2006.

[2] J. Caballero, C. Grier, C. Kreibich, and V. Paxson. Measuring pay-per-install: The commoditization of malware distribution. In *USENIX Conference on Security*, 2011.

[3] M. Egele, T. Scholte, E. Kirda, and C. Kruegel. A survey on automated dynamic malware-analysis techniques and tools. *ACM Comput. Surv.*, 44(2), 2012.

[4] L. Invernizzi and P. M. Comparetti. Evilseed: A guided approach to finding malicious web pages. In *IEEE Security and Privacy*, 2012.

[5] L. Invernizzi, S. Miskovic, R. Torres, C. Kruegel, S. Saha, G. Vigna, S. Lee, and M. Mellia. NAZCA: Detecting malware distribution in large-scale networks. In *NDSS*, 2014.

[6] J. Jang, D. Brumley, and S. Venkataraman. Bitshred: Feature hashing malware for scalable triage and semantic analysis. In *ACM CCS*, pages 309–320, 2011.

[7] A. Kapravelos, Y. Shoshitaishvili, M. Cova, C. Kruegel, and G. Vigna. Revolver: An automated approach to the detection of evasive web-based malware. In *USENIX Security*, 2013.

[8] D. Kirat, G. Vigna, and C. Kruegel. Barecloud: bare-metal analysis-based evasive malware detection. In *USENIX Security*, 2014.

[9] B. J. Kwon, J. Mondal, J. Jang, L. Bilge, and T. Dumitras. The dropper effect: Insights into malware distribution with downloader graph analytics. In *ACM CCS*, pages 1118–1129, 2015.

[10] Z. Li, S. Alrwais, Y. Xie, F. Yu, and X. Wang. Finding the linchpins of the dark web: A study on topologically dedicated hosts on malicious web infrastructures. In *IEEE Security and Privacy*, 2013.

[11] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying suspicious urls: An application of large-scale online learning. In *ICML*, 2009.

[12] T. Nelms, R. Perdisci, M. Antonakakis, and M. Ahamad. Webwitness: Investigating, categorizing, and mitigating malware download paths. In *USENIX Security*, 2015.

[13] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *NIPS*, 10(3):61–74, 1999.

[14] C. Rossow, C. Dietrich, and H. Bos. Large-scale analysis of malware downloaders. In *DIMVA*. 2013.

[15] D. Song, D. Brumley, H. Yin, J. Caballero, I. Jager, M. G. Kang, Z. Liang, J. Newsome, P. Poosankam, and P. Saxena. Bitblaze: A new approach to computer security via binary analysis. In *Information systems security*. 2008.

[16] A. Tamersoy, K. Roundy, and D. H. Chau. Guilt by association: Large scale malware detection by mining file-relation graphs. In *SIGKDD*, 2014.

[17] Y. Yamaguchi, C. Faloutsos, and H. Kitagawa. Socnl: Bayesian label propagation with confidence. In *PAKDD*, 2015.

[18] J. Zhang, C. Seifert, J. W. Stokes, and W. Lee. Arrow: Generating signatures to detect drive-by downloads. In *WWW*, 2011.

[19] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.