Guilt-by-Association: Detecting Malicious Entities via Graph Mining

Pejman Najafi(✉), Andrey Sapegin, Feng Cheng, and Christoph Meinel

Hasso Plattner Institute (HPI), Prof.-Dr.-Helmert-Straße 2-3,
14482 Potsdam, Germany
{pejman.najafi,andrey.sapegin,feng.cheng,christoph.meinel}@hpi.de

Abstract. In this paper, we tackle the problem of detecting malicious domains and IP addresses using graph inference. In this regard, we mine proxy and DNS logs to construct an undirected graph in which vertices represent domain and IP address nodes, and the edges represent relationships describing an association between those nodes. More specifically, we investigate three main relationships: *subdomainOf*, *referredTo*, and *resolvedTo*. We show that by providing minimal ground truth information, it is possible to estimate the marginal probability of a domain or IP node being malicious based on its association with other malicious nodes. This is achieved by adopting belief propagation, i.e., an efficient and popular inference algorithm used in probabilistic graphical models. We have implemented our system in Apache Spark and evaluated using one day of proxy and DNS logs collected from a global enterprise spanning over 2 *terabytes* of disk space. In this regard, we show that our approach is not only efficient but also capable of achieving high detection rate (96% TPR) with reasonably low false positive rates (8% FPR). Furthermore, it is also capable of fixing errors in the ground truth as well as identifying previously unknown malicious domains and IP addresses. Our proposal can be adopted by enterprises to increase both the quality and the quantity of their threat intelligence and blacklists using only proxy and DNS logs.

Keywords: Belief propagation · Big data analysis for security
Graph inference · Malicious domain and IP detection
Guilt-by-association · Graph mining

1 Introduction

In the case of both targeted threats (e.g., social engineering, spear-phishing, Advanced Persistent Threats, etc.) and mainstream threats (e.g., drive-by download, exploit-kits, malvertising, etc.), there exists an external malicious entity administered by an adversary that successfully reaches the end client (victim). The ability to block these entities from reaching the end client is considered to be an optimum cyber security solution. That's why so many organizations heavily invest in blocking these by deploying various security solutions such as

X. Lin et al. (Eds.): SecureComm 2017, LNICST 238, pp. 88–107, 2018.
https://doi.org/10.1007/978-3-319-78813-5_5

web application firewalls, proxy servers, email and web security appliance, etc. The majority of these solutions try to detect the maliciousness by analyzing the local features of those entities (e.g., URL structure or content of a web page). However, the problem is that the majority of these features are volatile, hence giving an advantage to cybercriminals to evade detection.

Consider Exploit-Kits (EK), e.g., Angler and Neutrino [11,14,29]. At high level internet users are first deceived to visit a previously compromised domain using techniques such as malvertising or malicious iFrames. Next using techniques such as HTTP POST redirection, domain generation algorithm (DGA), and HTTP redirects (302 cushioning) the victims are passed through various gateway pages to finally get to the landing page of the exploit kit. Next the EK tries to identify any potential vulnerabilities within the visitor's browser and plugins. Lastly, upon successful exploitation of a vulnerability, the client browser is silently forced to either download and run a malicious payload (Drive-by downloads) or execute shellcodes.

These tools go above and beyond to make it extremely difficult to detect the malicious entities involved (i.e., domains and IP addresses). For instance, to evade blacklisting exploit-kits use techniques such as domain shadowing [11], fast fluxing domains [10], and domain generation algorithm (DGA). In order to evade static and dynamic analysis of code or content, they use various anti-emulation, anti-sandbox, obfuscation and encoding techniques and dynamically build unique content and code for each request. In this regard, neither the maintenance of blacklists nor dynamic/static analysis of web pages is effective. This is due to the fact that there is no guarantee that next time the same landing page would have any local features shared with the previous observation of the landing page (i.e., different domain name, IP address, content, code, URL structure, etc). However, despite the fact that these tools are capable of mutating the entire local features, it is extremely challenging and sometimes costly to change global features (i.e., attributes shared between different malicious entities), for instance, the authoritative domain responsible for serving fast fluxing domains, the paths leading to two different landing page, or the registrar information.

This observation is not limited to exploit kits. Investigating the correlation between the global features of the previously known indicators of compromise (IOCs) could potentially allow us to better reason about new entities. In this paper, we formulate big data analysis for threat detection as a graph inference problem, with the intuition that malicious entities tend to have homophilic relationships with other malicious entities. More specifically, we focus on the analysis of proxy and DNS logs for the purpose of detecting malicious IP addresses and Domain names based on the relationships observed in those logs and minimal prior knowledge collected from threat intelligence (TI) sources. We achieve this by adopting Loopy Belief Propagation from probabilistic graphical models which allows to propagate the labels from labeled data to unlabeled data using relationships extracted from proxy and DNS logs.

1.1 Contribution and Road Map

The contributions of this paper are as follows:

- Proposal of an alternative approach to proxy and DNS log analysis for the purpose of threat detection using only the information available in those logs. This approach could be used by enterprises to increase both the quality and the quantity of their threat intelligence and blacklists.
- The successful adaptation of belief propagation as a graph based inference algorithm to propagate malicious labels using minimal ground truth to detect other malicious domains and IP addresses.
- Evaluation of the proposed approach on one day of proxy and DNS logs collected from a global enterprise, and demonstrating its capability to correct the inaccuracy in the original ground truth but also detect previously unknown malicious domains and IP addresses.

The rest of this paper is organized as follows. First, we provide some necessary background information while covering the most influential literature. In Sect. 3, we introduce our approach for detecting malicious entities. Section 4 provides an overview of our implementation. Section 5 describes our dataset and the experimental setup. Section 6 focuses on the evaluation and discussion. Section 7 discusses the limitation of our work and the potential directions for the future work, and finally, we conclude this paper in Sect. 8.

2 Background and Related Work

2.1 Proxy and DNS Logs

Nowadays, the majority of organizations, collect and store event logs generated by different components in the organization's premises such as firewalls, operating systems, proxy and DNS servers. Although traditionally the primary usage of these event logs was troubleshooting problems, nowadays they are collected due to mostly regulatory compliances and posthoc analysis. Two most valuable sources of event logs collected by many enterprises are DNS and proxy logs. While grasping the functionalities of proxy and DNS servers are beyond the scope of this paper, we will briefly cover the value of the logged events in the context of security analytics, and we would like to refer the reader to [13,22,23] to learn more about proxy servers and Domain Name System (DNS) respectively.

Due to the fact that web traffic is typically allowed by most of firewalls, HTTP, HTTPS, and DNS traffic is extensively abused by cybercriminals to reach the end users (e.g drive-by download, phishing website, bots communication with command-and-control servers, infrastructure management using fast fluxing [10], etc.), hence leading to the popularity of proxy and DNS log analysis in the security domain.

In this regard, Manners [19] discusses how it is possible to detect malicious entities based on abnormal or rare user agent string. Oprea et al. [25] address

the problem of detecting suspicious domains (associated with C & C) using features extractable from proxy logs. These features include domain connectivity (the number of hosts contacting to a domain), the referrer string, the user-agent string, access time correlation (domain visited by the same client within a relatively short time period), and IP space proximity. Ma et al. [16,17] discuss the detection of malicious websites based on lexical and host-based features of their URL (e.g., number of dots) with the intuition that malicious URLs exhibit certain common distinguishing features. Zhang et al. [36] use term frequency/inverse document frequency (TF-IDF) algorithm to tackle malicious URL detection, and Zhao et al. in [37] tackle the same problem using Cost-Sensitive Online Active Learning (CSOAL).

One of the first studies that explore DNS traffic analysis is [31] which proposes the construction of a passive DNS database by aggregating and collecting all unique, and successfully resolved DNS queries. This database is widely referred to as pDNS and is greatly adopted and researched in the security community. In this regard, Bilge et al. [3] introduce EXPOSURE, a system that employs large-scale, passive DNS analysis to detect malicious domains using features such as the number of distinct IP addresses per domain, number of the domains sharing an IP, average TLL, the percentage of numerical characters. In a similar research, Antonakakis et al. [1], propose Notos by focusing on the detection of agile malicious usage of DNS (e.g., fast-flux, disposable domains) using pDNS analysis. Notos distinguish itself by not only analyzing those features used in EXPOSURE, but also harvesting and analyzing complementary information such as the registration, DNS zones, BGP prefixes, and AS information. Later Antonakakis et al. [2] propose another system called Kopise. In contrast to Notos and EXPOSURE which analyze the traffic captured from local recursive DNS servers, Kopis monitor the traffic at the upper level of DNS hierarchy which pose its advantages and disadvantages. Perdisci et al. [27] investigate the detection of malicious flux service networks, and Yadav et al. address the problem of detecting algorithmically generated domain names used in domain and IP fluxes by looking at distribution of alphanumeric characters as well as bigrams of domains that are mapped to the same set of IP-addresses [34].

Our approach is fundamentally different to those mentioned above as those mostly target local features presented in proxy and DNS logs whereas we are interested in global features. We focus on identifying connected malicious entities based on their traces presented in those logs. We shall discuss our approach further in Sect. 3.

2.2 Graph-Based Inference

Inference refers to the process of reasoning about a variable based on a set of observations and evidence related to that variable. In this regard, graphs are ideal for capturing the correlation and dependency among different variables. That is the main reason why graph-based inference has been widely adopted in different research areas to tackle various inference problem with the intuition that neighboring nodes influence each other and this influence can be either

homophily (i.e., nearby nodes should have similar labels) or heterophily (i.e., nearby nodes should have different labels).

The most notable and popular graph based inference techniques are *graph-based Semi-Supervised Learning*, *Random Walk with Restart*, and *Belief Propagation*.

Random Walk with Restarts (RWR) is initially introduced as the underlying algorithm for Google's famous PageRank [4]. At high level, PageRank algorithm uses link information to assign global *importance* scores to all pages on the web (a web page can be considered as important if other important pages point to it). Similarly, TrustRank introduced by [9] adopts PageRank with different random walks to detect web spams by propagating *trust* label rather than importance. Other researchers have also built on top of TrustRank with minor changes introducing Distrust Rank and SybilRank [5,32].

Semi-Supervised Learning (SSL) techniques adopt the inference problem in machine learning to utilize unlabeled data with the intuition that similar data points connected by edges which represent their similarity should have same labels (also known as label propagation). Zhu et al. [38] introduce one of the pioneer works in graph-based SSL. The authors formulate the learning problem in terms of a gaussian random field on a graph in which vertices are labeled and unlabeled data and the edges are weighted similarities between vertices. Understanding the graph-based SSL is beyond the scope of this paper, to learn more, we refer the reader to [39].

Finally, Belief Propagation (BP) originally proposed by Pearl [26] also known as sum-product is one of the most efficient and popular inference algorithm used in probabilistic graphical models such as Bayesian Networks and Markov Random Fields. BP has been successfully applied to various domains such as image restoration [8], error-correcting [21], fraud detection, and malware detection [6].

For the purpose of this research, we have adopted BP due to its scalability and its success in other fields. In this regard, although there are still several important literature to cover for other graph inference techniques, the rest of this paper we will focus only on BP and the most influential literature that have adopted BP for the purpose of threat detection. To better understand the difference between the described graph-based inference techniques we would like to refer the reader to [15] where Koutra et al. compare *graph-based Semi-Supervised Learning*, *Random Walk with Restart*, and *Belief Propagation*, as this task is also beyond the scope of this paper and perhaps left to our future work.

2.3 Belief Propagation

Marginal probability estimation in graphs is known to be NP-complete, however, belief propagation provides a fast approximate technique to estimate marginal probabilities with time complexity and space complexity linear to the number of edges in a graph.

At the high level, BP infers a node's label from some prior knowledge about that node and other neighboring nodes by iteratively passing messages between all pairs of nodes in the graph. In this regard, in each iteration t every node i

generates its outgoing messages based on its incoming messages from neighbors in iteration $t-1$. Given that all messages are passed in every iteration, the order of passing can be arbitrary.

Let m_{ij} denote the message sent from i to j which intuitively represents i's opinion about j's likelihood of being in state x_j. This message is a vector of messages for each possible class, i.e., m_{ij} ($x_j = malicious$) and m_{ij} ($x_j = benign$). Mathematical determined as follows:

$$m_{ij}(x_j) \leftarrow \sum_{x_i \in X} \phi(x_i)\, \psi_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus j} m_{ki}(x_i) \tag{1}$$

where $N(i)$ is the set of nodes neighboring node i, and $\psi(x_i, x_j)$ is the *edge potential* which indicates the probability of a node i being in class x_i given that its neighbor j is in class x_j. $\phi(x_i)$ is called the *node potential function* which denotes the prior knowledge about a node, i.e., the prior probability of node i being in each possible class (in our case malicious and benign classes). And x_i represents a state from state space X.

The message passing phase terminates when messages do not change significantly between iterations, i.e., given a similarity threshold, the difference between the message sent from node i to node j at the iteration t and $t-1$ is less than the threshold, or when the algorithm reaches a predefined maximum number of iteration. At the end, each node will calculate its belief which is an estimated marginal probability, or formally $b_i(x_i)(\approx P(x_i))$ which represented the likelihood of random variable X_i to take value $x_i \in \{x_{mal}, x_{ben}\}$ determined as follows:

$$b_i(x_i) = k\phi(x_i) \prod_{x_j \in N(i)} m_{ji}(x_i) \tag{2}$$

where k is a normalizing constant to ensure the node's beliefs add up to 1 [35].

The original belief propagation algorithm proposed by Pearl [26] was designed to operate on singly connected networks (tree-structured graphical models), and provides an exact inference with all nodes' beliefs converging to the correct marginal in a number of iterations equal to the diameter of the graph (at most the length of the longest path in the graph). Although the presence of loops will cause the messages to circulate indefinitely hence not allowing the convergence to a stable equilibrium, it is possible to apply the algorithm to arbitrary graphical models by ignoring the presence of any potential cycles in the graph. This is typically referred to as loopy belief propagation (LBP) [24]. The convergence of loopy belief propagation is not guaranteed and the results are considered to be approximate, however, in practice, it often arrives at a reasonable approximation to the correct marginal distribution.

We shall later describe how we have adopted and tailored LBP to incorporate our domain knowledge which through the remainder of this paper will be referred to as BP algorithm.

2.4 Threat Detection via Belief Propagation

Malware detection is one of the areas that have successfully adopted BP. In this regard Polonium [6] is one of the first works tackling the problem of malware detection using large-scale graph inference with the intuition that good applications are typically used by many users, whereas, unknown (i.e., potentially malicious) applications tend to only appear on few computers. To test this hypothesis, the authors generated an undirected, unweighted bipartite machine-file graph, with almost 1 billion nodes and 37 billion edges. The graph vertices are of two types: machine and file vertices and the edges indicate the observation of a file on a machine. Polonium was not only evaluated using a prepared validation set but also tested in the-field by Symantec. In this regard according to Symantec's experts, Polonium has significantly lifted the detection rate by 10 absolute percentage points while maintaining 1% false positive rate when compared to other existing methods. This is arguably one of the most successful research adaptation of BP and graph inference in the security domain showing the potentials of this approach.

In a similar research, Tamersoy et al. [30] propose Aesop, a very similar system, that tackles the same problem. In this regard, Aesop utilizes locality sensitive hashing to measure the similarity between files to eventually construct a file to file graph to infer the files' goodness based on belief propagation. while Polonium is more concerned with the observation of malicious files on malicious a machine (i.e., file to machine relationship), Aesop is concerned with the similarity of files (file to file relationship).

Manadhata et al. [18] adopt BP and graph inference to detect malicious domains using enterprise's HTTP proxy logs. This is achieved by running BP on a host-domain graph which captures the enterprise's host connection to external domains. The authors estimate the marginal probability of a domain being malicious based on minimal ground truth. The intuition in this research is that infected hosts are more likely to visit various malicious domains whereas user behavior on benign hosts should result in benign domain access. The authors run BP on a constructed host-domain graph showing their approach capability to classify malicious domains with 95.2% TPR with a 0.68% FPR using 1.45% ground truth (blacklisted and whitelisted entities).

Our approach described in the next section is very similar to the one described by Manadhata et al., however, while Manadhata et al. are interested in machine to domain relationship, we are interested in domain to domain, domain to IP, and IP to IP relationships. More specifically while they mine proxy logs to construct a graph based on the relationship between internal entities and external entities (i.e., connections from client machine to external domains), we mine both proxy and DNS logs to construct a graph based on the relationships between external entities themselves (e.g., domain resolving to an IP address, or domain name referring to another domain name).

Other similar research includes [40] which takes a similar approach to [18] while focusing on DNS logs rather than proxy logs. In his regard, the authors focus on three main relationships extractable from DNS logs: (1) *connectsTo*

which indicates enterprise's host connected to a domain, (2) *resolvedTo* (DNS record type A) which indicates a domain resolving to an IPv4 address, and (3) CNAME which indicates a domain being an alias for another domain. [12] investigates the connection between domain, IP and URL. Oprea et al. [25] address early-stage APT detection using BP on host-domain graph extracted from proxy logs. And Rahbarinia et al. [28] propose Segugio to detect new malware-control domains based on DNS traffic analysis with a very similar intuition to [18].

3 Our Approach

3.1 Problem Description

Formally we formulate our inference problem as follows:

Given:

- An undirected graph $G = (V, E)$ where V corresponds to the collection of domain names and IP addresses, and E corresponds to the set of relationships between those domain and IP nodes. V and E are extracted from events in proxy and DNS Logs.
- Binay class labels $X \in \{x_{mal}, x_{ben}\}$ defined over V, where x_{mal} represents malicious label, and $P(x_{mal})$ the probability of belonging to class malicious. Note that $P(x_{mal})$ and $P(x_{ben})$ sums to one.

Find: The marginal probability $P(X_i = x_{mal})$, i.e., the probability of node i belonging to class malicious.

3.2 Graph Construction

The graph $G = (V, E)$ is constructed from events in the proxy and DNS logs.

The set of vertices V consists of two types of nodes: *domain names* and *IP addresses*. Domain names are valid parts of a fully qualified domain name (FQDN) excluding the top-level domain (TLD). For instance, considering x.example.com as a given FQDN, we then take example.com as the second-level domain and x.example.com as the third-level domain. Domain names are extracted from destination URLs in proxy logs, and the query section of A records presented in DNS logs. IP addresses are validated IP version 4 addresses observed in DNS logs (A records), and occasionally in proxy logs (sometimes the URL contains an IP address rather than a FQDN, also some proxy servers log the resolved IP address).

The set of edges E expresses three distinct relationships: *subdomainOf*, *referredTo*, and *resolvedTo*. In this regard, the subDomainOf relationship captures the dependency between different level of a FQDN, e.g., x.example.com is a subDomainOf example.com. This relationship is extracted from any valid FQDN logged in DNS or proxy logs. referredTo captures the connection between two domain/IP nodes if one has referred to the other one. This feature is extracted

from the referer field in HTTP request-header logged in proxy logs. And finally, resolvedTo captures the DNS resolution of a domain name to an IPv4 address, which is presented in DNS logs and occasionally in proxy logs. Figure 1 shows the graph constructed from raw events presented in DNS and proxy logs, and illustrates different type of nodes and relationships used in graph G.

Fig. 1. Domain-ip graph constructed from a sample of raw events in DNS and proxy logs showing the association between domain and IP nodes using *subdomainOf*, *referredTo*, and *resolvedTo* relationships

Our Intuition for These Three Relationships is: First, the usage of subdomains is one of the simplest yet effective techniques used by cyber criminals (e.g., DGA and domain shadowing) to evade blacklisting. Intuitively, different levels of a FQDN should belong to the same class. For instance, if x.example.com is listed as a malicious entity by a threat intelligence feed, it is likely that example.com and any other k-level domain under example.com (e.g., y.example.com) is also malicious. Second, the majority of the malware serving networks are composed of a tree-like structure in which the victims are usually redirected through various hops before landing on the main distribution site [20]. Although different victims might land on totally different sites, the redirection paths are usually overlapped. Furthermore, the HTTP referrer is also set while a domain (e.g., a website) is loading its modules from potentially different servers, therefore, indicating association among different domains. In this regard, the HTTP referrer can be used to infer the probability of a node being malicious based on the neighboring malicious nodes that have referred to it or it has referred to. One could also expand the referrer list by implying "referring" based on the correlation among different requests presented in proxy logs (e.g., requests from the same client in a short period of time) And finally, if a domain is listed as a malicious, intuitively we could assume that the resolved IPv4 address of that domain should also be labeled malicious at least for the duration of that resolution and vice versa.

3.3 Adaptation of BP

In this section, we describe our adaptation of BP algorithm described in Sect. 2 while incorporating domain knowledge, ground truth, and relationship weights.

Node Potential: As previously explained the node potential represents the prior knowledge about the state of each node. In this regard, we will assign different node potential to domain and IP nodes based on the ground truth. For example, we assign a prior $P(X_i = x_{mal}) = 0.99$ to the node i, if i is presented in the collected malicious domain/IP list, or assigning $P(X_i = x_{mal}) = P(X_i = x_{ben}) = 0.5$ for the nodes that are neither in the malicious list nor in the benign list (i.e., they are equally likely to be malicious or benign). Note that we avoid assigning a probability of 1 to any nodes to account for possible errors in the ground truth. Table 1 shows the node potentials assigned to each vertex on the graph, based on the prior knowledge (belief).

Table 1. Node potential based on the original state

Node	P (malicious)	P (benign)
Malicious	0.99	0.01
Benign	0.01	0.99
Unknown	0.5	0.5

Edge Potential: We will adjust the edge potential matrices to capture the intuition that neighboring nodes are more likely to have the same state due to a homophilic relationship.

Moreover, due to the fact that our graph consists of three unique edge types (referredTo, resolvedTo, subDomainOf) it is important to introduce a way to incorporate edge weight (importance). For example, two neighboring nodes that are connected via resolvedTo relationship should influence each other more than two nodes that are connected via referredTo. This edge weight is also incorporated in the edge potential. Table 2 shows the adjusted edge potential matrices.

Table 2. Edge potentials matrices

$\psi_{ij}(x_i, x_j)$	$x_j = benign$	$x_j = malicious$
$x_i = benign$	$0.5 + w\epsilon$	$0.5 - w\epsilon$
$x_i = malicious$	$0.5 - w\epsilon$	$0.5 + w\epsilon$

We experimented with different edge potential (adjusting ϵ) and although we noticed changes in final probability distributions, the end results were comparable as long as the weights captured the importance of different relationships. After experimenting with different w and ϵ and we appointed them as follows: $w_{referredTo} = 0.5$, $w_{resolvedTo} = 1.5$, $w_{subdomainOf} = 1.5$, and $\epsilon = 0.1$. It is worth to mention that although the edge weights seem trivial in this research, they will have a much higher impact when adding more edge types, therefore we will investigate them further in our future work.

Message Passing. There are two variants of message updating and passing protocol in BP: *asynchronous* and *synchronous*. At high level, in asynchronous BP, also known as sequential updating scheme, messages are updated and passed one at a time, whereas, in synchronous BP, also known as parallel updating scheme, messages are updated and passed in parallel.

Although BP is computationally efficient, i.e., the running time scales linearly with the number of edges in the graph, it is not sufficient to run it in the asynchronous mode for a graph of billion nodes and edges. Therefore, for large graphs, it is crucial to adopt parallel BP which can utilize multi-core architecture and parallelly execute the message updates and beliefs calculations.

4 Implementation

Due to the scalability requirements, we have implemented the BP algorithm with parallel updating scheme using Apache Spark framework. Apache Spark, the successor to Hadoop MapReduce is one of Apache's open-source projects that has gained so much momentum in both industry and academic due to its power of handling big data analytics. We have not only implemented the BP algorithm in Spark but also the *extract*, *transform*, *load* (ETL) modules as well as the graph itself. This design makes it possible to not only scale up (i.e., take advantage of more powerful hardware) but also scale out (i.e., distributing all modules to different machines).

Our implemented system is composed of five modules: (1) *Extraction*, (2) *Transformation*, (3) *Ground Truth Construction* (GTC), (4) *Loading*, and (5) *BP* as shown in Fig. 2. In summary, the extraction module, preprocesses the DNS and proxy logs by extracting, parsing, and validating the fields of interest as described in the previous sections. Then the transformation module converts the extracted values into unique vertices and edges. The ground truth construction (GTC) module is responsible for combining and adjusting the collected list of malicious/benign domain and IPv4 addresses, removing duplicate and the unmatched entities (malicious and benign entities that are in ground truth but not observed in the event logs). This module is also responsible for carefully selecting the validation set. The loading module receives the output of the transformation and GTC modules to construct a property graph and labeling each vertex based on ground truth (malicious, benign, unknown). And finally, BP module converts the constructed graph to Markov Random Field with the provided node and edge potentials, then runs the implemented BP algorithm to compute the beliefs following the procedure described in the previous section. In order to avoid numerical underflow (zeroing-out values), the whole math performed by BP module is carried in the log domain.

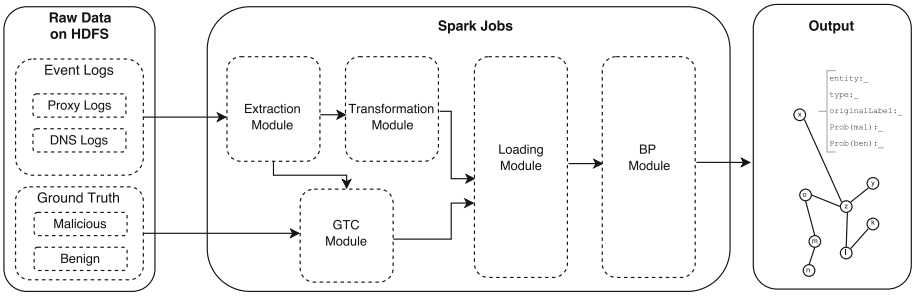

Fig. 2. System architecture diagram

5 Experimental Setup

5.1 Dataset Description

For the purpose of this research, we had access to one-day proxy and DNS logs collected from a large global enterprise. More specifically, 0.74 TB (terabytes) of proxy logs and 1.2 TB of DNS logs containing DNS requests and responses.

There is a total of 0.91B (billion), and 0.35B events in DNS and proxy logs respectively. After running ETL modules on those events we were capable of extracting approximately 1.89M (million) unique vertices and 4.29M unique edges.

5.2 Ground Truth Description

To assign priors to domain and IP vertices, a ground truth set was prepared by collecting a list of known malicious domain and IP addresses from both a commercial threat intelligence platform and various freely available sources including (but not limited) to Google Safe Browsing, AlienVault Open Threat Exchange, malwaredomainlist.com, malwaredomains.com. Similarly, we obtained a list of known benign domains from Cisco Umbrella (top one million most popular domains).

Ultimately, we were capable of collecting approximately 1M (million) unique malicious domains, 1M unique malicious IP addresses, and 1M unique benign domains. Once we checked those against our event logs we had a total of 2.12K (thousand) matched malicious entities and 0.29M (million) matched benign entities. This large gap and bias in the ground truth are due to the fact that it is quite unlikely for the enterprise hosts to be massively infected, i.e., the domains visited by the client were more likely to be benign rather than malicious. Therefore we had to adjust the benign data set to hold a balance between malicious and benign entities.

5.3 Hardware Setup and Runtime

Due to the fact that the entire modules are implemented in Apache Spark, it is possible to run the proposed approach on any configuration of hardware. In this regard, for the purpose of this research, we ran our experiment on a Spark cluster configured with 28-core 2.00 GHz linux machine with 100 GB of RAM.

Although, investigating the efficiency and the performance of the modules is beyond the scope of this research, to get a grasp of its performance, using the dataset and the hardware described above, the Extraction, Transformation, Ground Truth Construction, and Loading Modules all together take almost 50 min. In other words, 50 min to read the raw proxy and DNS logs as well as the collected GT, preprocess them, and write a parquet file containing the prepared unique vertices and unique edges. Then the BP module takes almost 90 min to read that parquet file, construct the Markov network and run 7 iterations of the described BP.

As discussed before, since there is no guarantee of BP convergence, it is important to introduce a convergence threshold. During our experiments, we noticed that 7 to 10 iterations were sufficient to get a reliable estimate. That means after 10 iterations the difference between the messages sent from i to j in iteration t compared to $t-1$ was negligible. It is worth to mention that the definition of negligible (i.e., convergence threshold) must be proportional to the edge potential matrices. In this regard, trying to spot small convergence threshold while assigning large values to w, or ϵ will produce many unnecessary iterations. In our experiment we set it to 0.01. This is due to the fact that our choice for the edge potential metrics forced a high influence.

It also worth to mention that despite our effort to adopt various techniques and design patterns to increase the performance of our modules, we noticed some idle time in the BP module which we suspect is due to our setup. In this regard our hardware setup with 100 GB of memory seems to be insufficient to hold the whole graph while running the BP and therefore causing SWAPs, IOs, as well as heavy garbage collection. It is possible to greatly improve the above numbers using various techniques suggested by Apache Spark[1] to tune the performance even on the same hardware setup.

6 Results and Discussion

In this section, we describe the evaluation of our approach based on the data set, ground truth and the experimental setup described above.

6.1 Validation

As mentioned before, one the GTC module's tasks is to carefully select a list of samples for the purpose of validation. In this regard, after constructing the ground truth which consists of a balanced number of matched malicious and

[1] http://spark.apache.org/docs/latest/tuning.html.

benign entities (i.e., domains and IP addresses), the GTC module carefully marks n samples for *validation* and the rest for *training*. Then the BP module uses the training set to set up the priors and assigns an unknown prior to nodes marked for the validation.

This validation set must be chosen carefully as it is quite likely that a node presented in the validation set would have no path to any node presented in the training set therefore not allowing us to properly evaluate our system. This is due to the fact that it is quite rare to find two connected malicious entities both presented in a blacklist or a TI feed (e.g., finding a domain and its resolved IP address both listed in a retrievable list). The majority of these feeds, only list the malicious entities they have observed (e.g., the IP address of the phishing site, or the domain name for a malware hosting domain).

Hence, before taking the samples, we had to calculate the connected components in our GT and choose the validation samples from those. For example, if we take node i as a malicious node for validation, it must have a path to at least another malicious node within the constructed graph. Furthermore, to hold a balance between benign and malicious nodes, we took half of the samples from malicious connected components and the other half from benign connected components.

We present our detection capability as Receiver Operating Characteristic (ROC) plot as shown in Fig. 3. This is achieved by thresholding malicious belief of nodes presented in the validation set. For instance, given a threshold t, and a node i, if i's malicious belief, $P(X_i = x_{mal}) > t$, then we predict i as malicious; else benign. This prediction is then compared to the i's original label to determine this detection as false positive, true positive, false negative, or true negative. After repeating this procedure for all the nodes in the validation set, it is possible to compute FPR and TPR for a given threshold t. And finally, plot the ROC based on different selections of t in the range of [0,1].

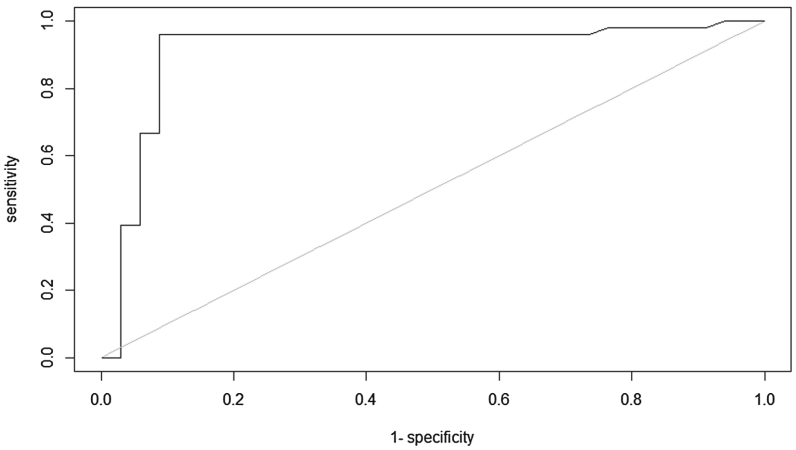

Fig. 3. Receiver Operating Characteristics (ROC) plot

As we can see the area under the ROC curve (AUC) is 91% (the higher the AUC, the better the classification), while we achieve 96% TPR with an 8% FPR.

6.2 Analysis of False Positives and False Negatives

To better understand the classification accuracy and the reason for the high FPR, we investigated the false positives (FPs) and the false negatives (FNs), i.e., benign domains that were wrongly classified as malicious and malicious entities that were wrongly classified as benign. We noticed the following observation when investigating these entities.

The majority of the FNs were entities associated with either cloud-based or online advertising services which introduce a significant challenge to our approach. For instance, we noticed that although some services were marked malicious in our ground truth, after running BP they were classified as benign due to their association with more benign entities. This decision making is reasonable as it does not fully make sense to mark an advertising platform completely malicious due to one malicious ad. Other FNs were malicious IP addresses that our system classified as benign, once investigated, we realized that although they may have been malicious at some point, that was no longer true. This classification can also be explained as these IP addresses were managed by cloud-based services that have reassigned those IP address.

FPs fell into two groups, first, entities that were classified wrongly due to a bad report from a TI source. For example, we noticed that there were some sub-domains that had their top-level domain wrongly blacklisted by Cisco Web Security Appliance[2] and therefore propagated to one of the threat intelligence sources we consumed, causing the subDomainOf relationship to overpower all the other referredTo relationships and eventually be classified as malicious. Second, entities that system correctly identified as malicious despite the fact that they were labeled benign in GT. For instance, we were able to identify 6 entities that had turned malicious just recently and the system was capable of detecting those based the referredTo relationship to other malicious entities.

In summary, the FNs and FPs were mostly the result of bad GT (inaccurate threat intelligence), and the attempt of the system to correct that inaccuracy. This investigation shows that although it is better to validate the crawled and consumed threat intelligence, it is not crucial as such system with more paths could potentially correct the TI inaccuracy.

6.3 Previously Unknown Malicious Entities

We also investigated the ability of our approach to detect new malicious entities (i.e., entities that were not presented in the ground truth). After running the BP, we selected the top 100 entities that were assigned a high probability of belonging to class malicious and did not exist in the GT.

[2] https://www.cisco.com/c/en/us/products/security/web-security-appliance/index.html.

Although validating these entities is a challenging task (i.e., how one decides maliciousness), for the purpose of this research we manually validated those entities and concluded maliciousness if we observed a reputable threat intelligence source (e.g., VirusTotal[3], URLVoid[4], AlienVault Open Threat Exchange[5]) reporting on that entity. It is also worth to mention that for this task, we discarded the indicator's time stamp (i.e., the time in which that entity was seen). This is due to the fact that, investigating and validating the maliciousness period, is itself an extremely challenging task.

74% of those entities were, in fact, malicious entities that did not exist in our GT, but have been reported as malicious by several other TI sources. These entities were mostly domains and IP addresses associated with services running on CloudFront[6]. 9% were entities that we could not find any information about. They seem to be either services that were active for a short period of time and detected to be malicious based on their previous association with malicious entities, or the result of DGA. However, we could not validate those intuitions as there was no trace of them on the internet. And the rest were entities that were classified wrongly.

In summary, investigating previously unknown malicious entities showed that our approach was capable of detecting new malicious entities that were not presented in our ground truth. And despite the fact that there were some misclassified entities (FPs), this approach is still extremely effective as blocking these FPs would not have a drastic effect, due to the fact that the main reason for classifying them as malicious, was, not being associated with major benign entities.

7 Future Work

One of the limitations of our work is the experimental setup. In this regard, we only had access to one day of proxy and DNS logs. It could be interesting to investigate how increasing the volume of the event logs will affect the detection. One could expect to have a better detection accuracy, as there will be more paths within the graph.

Next, there was a time gap of 6 months between our ground truth preparation and events collected by the enterprise's servers, i.e., the event logs were already 6 months old by the time we got access to them. This could potentially introduce various errors into our detection capabilities as usually threat intelligence feeds are time sensitive. It would be interesting to evaluate this system on live event logs and based on more accurate ground truth.

The next limitation of this system as in its current status is the fact that a malicious entity (e.g., a malicious domain) is capable of defeating the system by

[3] https://www.virustotal.com.
[4] http://www.urlvoid.com.
[5] https://otx.alienvault.com/.
[6] https://aws.amazon.com/cloudfront.

referring to many benign entities. Although it is quite unlikely that this is currently happening, to prevent such scenario, one could adjust the edge potential to only allow propagation in one direction. We plan to investigate this behavior in our future work.

Furthermore, in this paper, we only focused on three main relationships that directly connect domain names and IPv4 addresses. However, one could investigate indirect relationships such as nameServerFor, mailServerFor, aliasFor, with the intuition that cyber criminals tend to reuse infrastructure for their malicious entities. Additionally, it is also possible to take other host-based relationships into consideration (e.g., user agent, IP address, MAC address), thus enabling the propagation of the client machines' reputation to domains and IP addresses (a similar approach to [18,40]).

In addition, it is also possible to look into enriched relationships such as registrar information, IP ranges, ASN, and BPG in order to construct a larger graph with a higher connectivity. Antonakakis, et al. [1] make use of BGP, AS, and registration features as part of their feature set to detect malicious domains. [25] uses IP space proximity to measure the similarity between domains, [7,33] investigate features such as name servers, and registrant information to detect malicious domains. In this regard, the investigation of these combined relationships pose an interesting direction for the future work.

Finally, one could combine the global features (i.e., the features that would allow us to either directly or indirectly connect two entities) together with local features, such as, URL structure, port number, request/response length, and etc. This combination of global and local features should, in theory, improve the accuracy.

8 Conclusion

In this paper, we tackled the problem of detecting malicious domains and IP addresses by transforming it into a large-scale graph mining and inference problem. In this regard, we proposed an adaptation of belief propagation to infer maliciousness based on the concept of guilt-by-association using *subdomainOf*, *referredTo*, and *resolvedTo* relationships between IP and domain nodes. We evaluated our approach by running an adaptation of loopy belief propagation on a graph constructed from 2TB of proxy and DNS logs collected from a global enterprise. The results showed that our system attained a TPR of 96% at 8% FPR. While investigating the FP and FN we noticed the mistakes in the GT which was corrected by our system. We also investigated the system's ability to detect previously unknown malicious entities and demonstrated its capability to extend threat intelligence and blacklist by detecting new malicious entities.

References

1. Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., Feamster, N.: Building a dynamic reputation system for DNS. In: USENIX Security Symposium, pp. 273–290 (2010)
2. Antonakakis, M., Perdisci, R., Lee, W., Vasiloglou II, N., Dagon, D.: Detecting malware domains at the upper DNS hierarchy. In: USENIX Security Symposium, vol. 11, pp. 1–16 (2011)
3. Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M.: Exposure: finding malicious domains using passive DNS analysis. In: NDSS (2011)
4. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst. **30**(1), 107–117 (1998)
5. Cao, Q., Sirivianos, M., Yang, X., Pregueiro, T.: Aiding the detection of fake accounts in large scale social online services. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, p. 15. USENIX Association (2012)
6. Chau, D.H.P., Nachenberg, C., Wilhelm, J., Wright, A., Faloutsos, C.: Polonium: tera-scale graph mining and inference for malware detection. In: Proceedings of the 2011 SIAM International Conference on Data Mining, pp. 131–142. SIAM (2011)
7. Felegyhazi, M., Kreibich, C., Paxson, V.: On the potential of proactive domain blacklisting. LEET **10**, 6 (2010)
8. Freeman, W.T., Pasztor, E.C., Carmichael, O.T.: Learning low-level vision. Int. J. Comput. Vis. **40**(1), 25–47 (2000)
9. Gyöngyi, Z., Garcia-Molina, H., Pedersen, J.: Combating web spam with trustrank. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, vol. 30, pp. 576–587. VLDB Endowment (2004)
10. Holz, T., Gorecki, C., Rieck, K., Freiling, F.C.: Measuring and detecting fast-flux service networks. In: NDSS (2008)
11. Howard, F.: A closer look at the Angler exploit kit (2015). https://news.sophos.com/en-us/2015/07/21/a-closer-look-at-the-angler-exploit-kit/
12. Huang, Y., Greve, P.: Large scale graph mining for web reputation inference. In: 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1–6. IEEE (2015)
13. Scarfone, K.A., Hoffman, P.: Guidelines on firewalls and firewall policy (2009). https://www.nist.gov/publications/guidelines-firewalls-and-firewall-policy
14. Kotov, V., Massacci, F.: Anatomy of exploit kits. In: Jürjens, J., Livshits, B., Scandariato, R. (eds.) ESSoS 2013. LNCS, vol. 7781, pp. 181–196. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36563-8_13
15. Koutra, D., Ke, T.-Y., Kang, U., Chau, D.H.P., Pao, H.-K.K., Faloutsos, C.: Unifying guilt-by-association approaches: theorems and fast algorithms. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011. LNCS (LNAI), vol. 6912, pp. 245–260. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23783-6_16
16. Ma, J., Saul, L.K., Savage, S., Voelker, G.M.: Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1245–1254. ACM (2009)
17. Ma, J., Saul, L.K., Savage, S., Voelker, G.M.: Identifying suspicious URLs: an application of large-scale online learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 681–688. ACM (2009)

18. Manadhata, P.K., Yadav, S., Rao, P., Horne, W.: Detecting malicious domains via graph inference. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8712, pp. 1–18. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11203-9_1
19. Manners, D.: The user agent field: analyzing and detecting the abnormal or malicious in your organization (2011)
20. Mavrommatis, N.P.P., Monrose, M.A.R.F.: All your iframes point to us (2008)
21. McEliece, R.J., MacKay, D.J.C., Cheng, J.F.: Turbo decoding as an instance of pearl's "belief propagation" algorithm. IEEE J. Sel. Areas Commun. **16**(2), 140–152 (1998)
22. Mockapetris, P.: Domain names - concepts and facilities (1987). https://www.ietf.org/rfc/rfc1034.txt
23. Mockapetris, P.: Domain names - implementation and specification (1987). https://www.ietf.org/rfc/rfc1034.txt
24. Murphy, K.P., Weiss, Y., Jordan, M.I.: Loopy belief propagation for approximate inference: an empirical study. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, pp. 467–475. Morgan Kaufmann Publishers Inc. (1999)
25. Oprea, A., Li, Z., Yen, T.F., Chin, S.H., Alrwais, S.: Detection of early-stage enterprise infection by mining large-scale log data. In: 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 45–56. IEEE (2015)
26. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, Burlington (2014)
27. Perdisci, R., Corona, I., Dagon, D., Lee, W.: Detecting malicious flux service networks through passive analysis of recursive DNS traces. In: Annual Computer Security Applications Conference, ACSAC 2009, pp. 311–320. IEEE (2009)
28. Rahbarinia, B., Perdisci, R., Antonakakis, M.: Segugio: efficient behavior-based tracking of malware-control domains in large ISP networks. In: 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 403–414. IEEE (2015)
29. Rocha, L.: Neutrino exploit kit analysis and threat indicator (2016)
30. Tamersoy, A., Roundy, K., Chau, D.H.: Guilt by association: large scale malware detection by mining file-relation graphs. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1524–1533. ACM (2014)
31. Weimer, F.: Passive DNS replication. In: First Conference on Computer Security Incident, p. 98 (2005)
32. Wu, B., Goel, V., Davison, B.D.: Propagating trust and distrust to demote web spam. MTW **190** (2006)
33. Xu, W., Sanders, K., Zhang, Y.: We know it before you do: predicting malicious domains. In: Proceedings of the 2014 Virus Bulletin International Conference, pp. 73–77 (2014)
34. Yadav, S., Reddy, A.K.K., Reddy, A.N., Ranjan, S.: Detecting algorithmically generated domain-flux attacks with DNS traffic analysis. IEEE/ACM Trans. Netw. **20**(5), 1663–1677 (2012)
35. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Understanding belief propagation and its generalizations. Exploring Artif. Intell. New Millennium **8**, 236–239 (2003)
36. Zhang, Y., Hong, J.I., Cranor, L.F.: CANTINA: a content-based approach to detecting phishing web sites. In: Proceedings of the 16th International Conference on World Wide Web, pp. 639–648. ACM (2007)

37. Zhao, P., Hoi, S.C.: Cost-sensitive online active learning with application to malicious URL detection. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 919–927. ACM (2013)
38. Zhu, X., Ghahramani, Z., Lafferty, J., et al.: Semi-supervised learning using Gaussian fields and harmonic functions. ICML **3**, 912–919 (2003)
39. Zhu, X., Lafferty, J., Rosenfeld, R.: Semi-supervised learning with graphs. Carnegie Mellon University, Language Technologies Institute, School of Computer Science (2005)
40. Zou, F., Zhang, S., Rao, W., Yi, P.: Detecting malware based on DNS graph mining. Int. J. Distrib. Sens. Netw. (2015)