PROBABILISTIC THREAT PROPAGATION FOR MALICIOUS ACTIVITY DETECTION

Kevin M. Carter, Nwokedi Idika, and William W. Streilein

MIT Lincoln Laboratory, 244 Wood St., Lexington, MA 02420
{kevin.carter, nwokedi.idika, wws}@ll.mit.edu

ABSTRACT

In this paper, we present a method for detecting malicious activity within networks of interest. We leverage prior community detection work by propagating threat probabilities across graph nodes, given an initial set of known malicious nodes. We enhance prior work by employing constraints which remove the adverse effect of cyclic propagation that is a byproduct of current methods. We demonstrate the effectiveness of Probabilistic Threat Propagation on the task of detecting malicious web destinations.

***Index Terms*—** Graph algorithms, community detection, network security, blacklist

1. INTRODUCTION

In the domain of network security analysis, detection of malicious activity is of critical concern for network defenders. In many circumstances, network defenders have been able to identify nodes of known maliciousness through provided blacklists [1, 2]. While this strategy is a great tool for defending networks, it is limited by the things that are known to be malicious [3]. In many circumstances, domains appear on blacklists several days after the domain began serving malicious content. Using blacklisted sites as *tips,* analysis can be performed on network traffic to identify additional nodes associated with these known malicious nodes. Previous work has shown that malicious activity is highly localized in network spaces [4, 5], forming *communities* of maliciousness.

The task of identifying members of a group from a network topology has been proposed in several community detection algorithms [6]. Certain methods have leveraged modularity [7] applied to local structures with tip nodes both iteratively [8, 9] and through eigenspace analysis [10]. These methods require the communities of interest to be densely populated, which often is untrue for many applications producing sparse and/or bi-partite graphs.

Recent methods [11, 12, 13] take an iterative approach in which 'threat' is initialized at the tip nodes and iteratively propagated throughout the graph. These methods have proven promising, yet suffer from what we deem as *direct feedback*, in which a node's threat level can increase solely based on the threat it had previously propagated to neighbors. This is because the methods propagate threat without consideration of the provenance of the threat at each node. The light illustration in Fig. 1 demonstrates the false inference performed on node B because probability is fed back from node C.

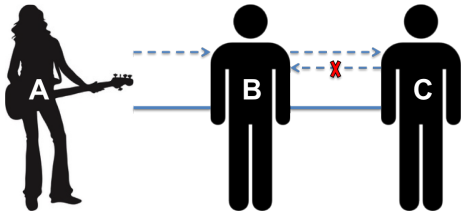

Fig. 1. Illustrative communication network with tip node A. Probability (dashed lines) propagates from $A \rightarrow B$, then from $B \rightarrow C$. Propagation from $C \rightarrow B$ is undesirable because $P(C)$ is entirely dependent on $P(B)$.

In this paper, we present a method for finding activity based communities in a graph given tips of known actors. Our method builds upon prior work [11] using an iterative threat propagation approach. However we avoid the issues of direct feedback by considering threat provenance; developing a method that yields probabilistic outputs rather than simple rank ordered lists. We refer to our method as *Probabilistic Threat Propagation* (PTP), and apply it towards the application of detecting malicious web domains and expanding blacklists. We show that it results in better performance than prior methods that suffer from direct feedback.

1.1. Related Work

Markov Random Fields (MRF) [14] model random variables as nodes in an undirected graph in which two non-neighboring nodes are conditionally independent from one another given the neighborhood of either node. When given an observation of a random variable, or a *tip*, the problem becomes a Conditional Random Field (CRF) [15]. Despite the assortment of traditional inference algorithms available for CRFs [16, 17], such algorithms are insufficiently general.

This work is sponsored by the Department of Defense under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

Specifically, these algorithms are applicable to CRFs only in cases where the interaction between adjacent nodes is the same irrespective of message flow direction. In applications where 'threat' is being passed between nodes, the flow direction should affect the beliefs about nodes in the graph. The tip nodes serve as the source of the threat flows. Without knowing the tips in advance, potential functions for a CRF cannot be defined to yield the desired results.

In attempts to remedy the issue of direct feedback, previous methods have employed a dampening factor [11] or edge weights [12] less than unity, which exponentially decrease the propagated threat level of any given node such that all connected nodes will not converge to the same value. As a byproduct of this dampening, however, the results are merely a rank-ordered list of threat levels that have no probabilistic meaning. In problem domains such as threat prioritization, a rank ordered list is sufficient. However, there are many scenarios in which an operator only wants an alert if some output score exceeds a threshold. This is difficult in prior methods as the scale is relatively meaningless except with respect to other nodes in the graph; this can continuously change as the graph topology evolves. Conversely, having an output measure of an actual probability makes both thresholding and explanation easier, and enables combining with outputs from other probabilistic detectors.

Previous methods have approached this task of identifying malicious domains by analyzing many features of the domain itself [18] or the activity observed by members of the same sub-network [5]. Our work is agnostic to both the domain features and activities, relying only on associations between domain name and IP address.

2. PROBABILISTIC THREAT PROPAGATION

For the purposes of this paper, we define two communities of interest for the detection problem: malicious and benign. We define the probability of being in the malicious community as $P(x)$, for which the probability of being in the benign community is $1-P(x)$; this notation intentionally allows 'partial' community membership. Given that this can be interpreted as the "threat level" of a particular node, we utilize the term interchangeably with "threat probability".

The intuition behind Probabilistic Threat Propagation (PTP) is that the level of threat at each node is equal to the weighted sum of the threat of neighboring nodes, discounting the level of threat those nodes received from the node of interest. Mathematically speaking, we define the graph $G = (X, E)$, where X represents the set of nodes and E is the set of edges representing some quantifiable *direct* relationship between nodes. We compute threat on node x_i as the probability of maliciousness, defined as:

$$P(x_i; G) = \sum_{j \in \mathcal{N}(x_i)} w_{ij} P(x_j | x_i = 0; G), \qquad (1)$$

where $\mathcal{N}(x_i)$ is the neighborhood of x_i – those nodes x_j such that $e_{ij} \in E$ – and w_{ij} is the weight of the edge e_{ij}[1]. Note that this formulation is very similar to [11]. The difference, while minor, is significant; by conditioning on the current node being equal to 0, we compute the probability of neighboring nodes in the absence of the node of interest. This guarantees that any increased probability on node x_j directly due to x_i will not in turn increase the probability of x_i. For ease of notation, we define $P(x_i) = P(x_i; G)$, as all probabilities are recursively defined through the parameterized graph G, with weightings w_{ij} pre-defined and incorporated into the full graph structure.

We note that graphs with cycles have the potential to increase threat due to indirect relationships stemming from a direct relationship with a tip node; an example of such is the 'square' $X = \{A, B, C, D\}$, $E = \{e_{AB}, e_{BC}, e_{BD}, e_{CD}\}$, and $\{tips\} = A$. Accounting for these types of cycles in general is not tractable due to the exponential increase in memory and message passing required, and in practice has minimal impact on the result due to the intrinsic exponentially reduced value of the threat passed.

Given that the probabilities are generally unknown, we solve Eq. (1) iteratively. We initialize PTP with the set $\{tips\}$, those nodes which are known to be malicious, and assign them *a priori* probabilities $P(x \in \{tips\}) = \gamma$, where $\gamma \in [0, 1]$ relays the confidence in the tip. All other nodes are initialized to priors of $P(x \notin \{tips\}) = 0$. At each iteration, we compute the new probabilities using Eq. (1), and reassign $P(x \in \{tips\}) = \gamma$, such that tip nodes remain constant. This process continues until convergence of $P(x_i), \forall i$.

2.1. Approximate Inference

Exact inference of Eq. (1) requires $P(x_j | x_i = 0)$ be computed for all pairs of nodes in the graph, an $O(N^2)$ operation which is generally intractable for large graphs. This intractability is exacerbated when the graph G contains cycles, which is generally the case for cyber networks. Hence, we develop an efficient approximation to Eq. (1) which leverages our recursively defined probabilities. As opposed to computing the weighted sum of conditional probabilities, we compute the weighted sum of the marginal probabilities $P(x_j)$, and subtract the contribution that node x_i had on node x_j in the previous iteration. Mathematically, we now solve iteratively over k:

$$P^k(x_i) = \sum_{j \in \mathcal{N}(x_i)} w_{ij} (P^{k-1}(x_j) - C^{k-1}(x_i, x_j)), \qquad (2)$$

where $P^k(x)$ is the probability of x at iteration k, and $C^{k-1}(x_i, x_j)$ is the portion of $P^{k-1}(x_j)$ which was directly computed from x_i in the previous iteration. This achieves

[1] We utilize undirected graphs such that $e_{ij} \in E \implies e_{ji} \in E$.

Algorithm 1 Probabilistic Threat Propagation

Require: $W, \{tips\}, \gamma$
  $P \leftarrow 0^N, P(\{tips\}) \leftarrow \gamma, C \leftarrow 0^{N \times N}$
  repeat
    $T \leftarrow W \otimes P^T$
    $C \leftarrow T - W \circ C^T$
    $P \leftarrow < C, \bar{1} >$
    $C(\{tips\}, \cdot) \leftarrow 0$
    $P(\{tips\}) \leftarrow \gamma$
  until P has converged
  return P

our desire of removing the direct influence x_i has on x_j from feeding back to x_i.

We are able to efficiently solve Eq. (2) using linear algebra. Given a set of N nodes, let us define the node threat probability vector $P \in \mathcal{R}^N$ such that $P(i) = P(x_i)$, the weight matrix $W \in \mathcal{R}^{N \times N}$ such that $W(i,j) = w_{ij}$, the transfer matrix $T \in \mathcal{R}^{N \times N}$ such that $T(i,j) = W(i,j) * P(j)$, and the contribution matrix $C \in \mathcal{R}^{N \times N}$, which we initialize to zeros. At each iteration, we assign $C = T - W \circ C^T$, where $A \circ B$ denotes the Hadamard product (e.g. $A(i,j) * B(i,j)$). This equation is the interior of the sum in Eq. (2), which computes each node's contribution to each other node, subtracting off the contribution from the previous iteration. To compute the node probabilities, we simply sum the elements of the newly computed C,

$$P =< C, \bar{1} >, \tag{3}$$

where $\bar{1}$ is the N element vector of ones and $< \cdot, \cdot >$ represents the inner product. Algorithm 1 details the full computation performed by PTP; our empirical results on over 100k node graphs typically converge in less than 50 iterations and on the order of seconds with commodity hardware.

2.2. Assigning Weights

The weight matrix W can be computed generically via the function $w_{ij} = f(x_i, x_j)$. Specifically, the function $f(\cdot, \cdot)$ can be defined in several ways, depending on the desired properties of the system, which we generically describe as

$$f(x_i, x_j) = \frac{1}{\sum_k g(x_i, x_k)} g(x_i, x_j), e_{ij} \in E, \tag{4}$$

for some function $g(x_i, x_j)$, which measures the interaction between nodes x_i and x_j. For example, in a social network, $g(\cdot, \cdot)$ could measure the number of times two persons communicated. In [11], kernels are proposed as weighting functions based on the temporal aspects of the edge. In order to maintain the probabilistic nature of our algorithms, it is important to normalize such that $\sum_j f(x_i, x_{j)} = 1$. For the remainder of this paper, we define $g(x_i, x_j) = 1, \forall e_{ij} \in E$, yielding a final weight for each edge e_{ij} proportional to degree of node x_i.

3. MALICIOUS DOMAIN ANALYSIS

We now show that PTP can be used to detect malicious web domains, leveraging the fact that a single IP may host numerous malicious domains, and a single malicious domain may resolve to multiple IP addresses. However, web hosting services make it such that a single IP may host thousands of websites, with the vast majority of them being benign. Simply blacklisting any IP address that is associated with a malicious domain would yield an extremely high false positive rate.

For example, we observed one blacklisted site that resolved to the same web hosting IP address as 13 other sites, none of which were on the blacklist. These included sites such as HBAMA.COM, which is for the Home Builders Association of Massachusetts. Had we simply blacklisted the IP address, sites such as these would be unnecessarily blocked as well. On the other hand, the PTP output was $P(x) = 0.0714$, which is far below any reasonable threshold for alerting.

3.1. Data

We create a bipartite graph with edges connecting domains and IP addresses, specifically $e_{ij} \in E \iff$ 'domain i resolves to IP address j'. To populate this graph, we leveraged the web proxy logs obtained from a medium-sized enterprise network. The proxy server on this network employed the McAfee TrustedSource Web Database [19] to validate web requests and appropriately label some of these as *Trusted, Malicious,* or *Suspicious*. Using these labels, we create a blacklist (and whitelist) from 65 days of web requests, focusing specifically on the fully-qualified domain name (FQDN) of the URL, generating $\{tips\}$ such that $P(x \in Malicious) = 1$, $P(x \in Suspicious) = 0.5$, and $P(x \in Trusted) = 0$. This is to be used as the foreground graph (e.g. ground truth). We captured all traffic for the 65th day to use as background. Our final data set contains 77,813 domain nodes and 48,921 IP nodes, which contain the subset 2,100 and 2,186 blacklisted domains and associated IP addresses, respectively. We note that there is overlap between the blacklist and the captured background data.

3.2. Malicious Detection

The goal of this experiment is to determine, given a partial blacklist, how much of the full blacklist can be recovered while keeping false alarms low. Operationally, this is equivalent to expanding a blacklist given one that is available, helping to identify additional sources of malicious activity that have not yet been detected by the blacklist providers.

Over a 20-fold cross validation, we randomly select 50% of the blacklisted nodes as training tips, keeping their $P(x)$ constant, and select as a test set any domain that is reachable within the foreground graph (e.g. ground truth) from any of the tip nodes. The test set contained an average of 205

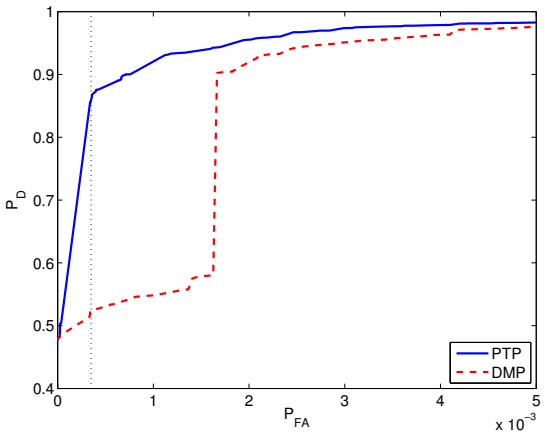

Fig. 2. Performance comparison of PTP and DMP for detecting malicious domains. The vertical line highlights the performance gap in the low P_{FA} regime.

nodes. After adding the training and test data to the background graph, we apply PTP and alert on any domain with $P(x) \geq \tau$, for some threshold τ. When computing detection probability, P_D, and false alarm probability, P_{FA}, any domain that is not in the blacklist is considered to be benign.

In Fig. 2 we plot Receiver Operating Characteristic (ROC) curve by varying τ, showing that PTP is able to detect 87% of the malicious domains with a false alarm rate of 0.035%. On average, that performance corresponds to 178 newly detected malicious domains and 30 false alarms for that day. As expected, PTP does a very good job of detecting malicious domains while generating very low false positives; this is due to the intuition that malicious infrastructure is shared.

For comparison, we ran the same experiment using dynamic membership propagation (DMP) [11], selecting parameters which yielded optimal performance. We plot the results in Fig. 2 as well, showing that PTP significantly outperforms DMP in the low false-positive regime. When limiting to the same $P_{FA} = 0.00035$, DMP was only able to detect 53.26% of the domains (109 on average). To maintain the same level of $P_D = 0.87$, running DMP would require a false alarm rate of $P_{FA} = 0.0017$, yielding an average of 129 false positives, nearly 100 more than PTP. Recall once again that the sole differentiating factor between PTP and DMP as parameterized is the direct feedback suffered by DMP. Hence, the modifications presented in this paper are indeed significant; not only does direct feedback produce an output that needs to be interpreted with respect to other nodes (e.g. rank-ordered list), the detection performance is negatively impacted as well. We note that the sharp increase in P_D observed in each method occurs when τ covers graphs with exactly 2 domains, one of which is in $\{tips\}$, mapping to the same IP (e.g. $\tau = 0.5$ for PTP).

3.3. Malicious Prediction

While the results presented in Fig. 2 are with respect to the *known* malicious domains, the fact remains that blacklists are imperfect observations. Many sites reach a blacklist long after they've been originally stood up and visited by unsuspecting victims. Indeed, the primary motivation behind this work is to identify malicious domains before they reach a blacklist. Hence, our false positive rate presented in previous experiments is an upper bound, as assuredly there is some chance that our 'false' detections are indeed malicious.

We now apply PTP on the same IP–domain graph as previously studied, only this time we utilize the entire blacklist as $\{tips\}$. We select the same operating point as in our previous experiment ($\tau = 0.49$), and detect 63 potentially malicious domains. Many detections were common typos, such as WWW.USATODYA.COM, WWW.YOTUBE.COM, and JETBLU.COM; these domains clearly are not false positives. This is a tactic used by malicious actors to obtain credentials or park a domain for ad revenue. Adding to the certainty is that a domain such as WWW.USATODYA.COM resolves to the same IP address as FIEDELITY.COM and AMERICANEXSPRESS.COM – both of which were on the blacklist.

Other detected sites are more difficult to assess from the FQDN, such as CHANNEL-SURVEY-CENTER.COM and WW35.MYSPORTSCLUB.COM. We note that both of these sites, as well as WWW.YOTUBE.COM and many other detections, ended up being blacklisted by the same service [19] weeks and even months after we were able to detect them with PTP. While anecdotal, this demonstrates the power of our method. During operation, an analyst can assess each detection and determine whether or not it should be added to the blacklist. This type of human-in-the-loop inference increases the fidelity of the model by adding more labeled data. This task becomes easier as time passes, as detections are likely to persist.

4. CONCLUSIONS

We have presented a graph analysis technique which determines the statistical probability that a node in an observed network is part of an activity-based community. PTP operates by taking tip nodes that are known to be in the community and associating observed relationships to obtain a probability value at each unknown node. We have shown that it is critical to consider threat provenance in this iterative process. We demonstrated the use of PTP by showing the ability to recover large portions of a blacklist and predict additional malicious domains, while yielding low false positives.

In future work, we will analyze different edge weighting functions [11] and see how our method fits with non-iterative solutions [13]. Finally, we will apply PTP towards non-malicious communities, propagating 'trust' as an example, or even multiple communities of interest.

5. REFERENCES

[1] Google, "Google SafeBrowsing API," http://code.google.com/apis/safebrowsing/, 2012.

[2] "URLBlacklist.com," http://www.urlblacklist.com, 2012.

[3] S. Sinha, M. Bailey, and F. Jahanian, "Shades of grey: On the effectiveness of reputation-based blacklists," in *Proceedings of the Intl. Conf. on Malicious and Unwanted Software (Malware)*, 2008, pp. 57–64.

[4] T. Yu, R. Lippmann, J. Riordan, and S. Boyer, "EMBER: A global perspective on extreme malicious behavior," in *Proceedings of the Seventh International Symposium on Visualization for Cyber Security*, 2010, pp. 1–12.

[5] M. P. Collins, T. J. Shimeall, S. Faber, J. Janies, R. Weaver, M. D. Shon, and J. Kadane, "Using uncleanliness to predict future botnet addresses," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, 2007, pp. 93–104.

[6] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3–5, pp. 75–174, 2010.

[7] M. Newman, *Networks: An Introduction*. Oxford University Press, 2010.

[8] A. Clauset, "Finding local community structure in networks," *Physical Review E*, vol. 72, 2005.

[9] U. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, 2007.

[10] B. Miller, M. Beard, and N. Bliss, "Eigenspace analysis for threat detection," in *Fusion*, 2011.

[11] S. Philips, E. K. Kao, M. Yee, and C. Anderson, "Detecting activity-based communities using dynamic membership propagation," in *Proceedings of the Intl. Conf. on Acoustics, Speech, and Signal Processing*, 2012.

[12] B. Coskun, S. Dietrich, and N. Memon, "Friends of an enemy: Identifying local members of peer-to-peer botnets using mutual contacts," in *Proceedings of 26th Annual Computer Security Applications Conference*, Dec. 2010, pp. 131–140.

[13] S. T. Smith, S. Philips, and E. K. Kao, "Harmonic space-time threat propagation for graph detection," in *Proceedings of the Intl. Conf. on Acoustics, Speech, and Signal Processing*.

[14] R. Kindermann and J. Snell, *Markov Random Fields and Their Applications*. American Mathematical Society, 1980.

[15] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML'01 Proceedings of the 8th International Conference on Machine Learning*, 2001, pp. 282–289.

[16] C. Sutton and A. McCallum, *Introduction to Conditional Random Fields for Relational Learning*. MIT Press, 2006.

[17] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[18] J. Ma, L. Saul, S. Savage, and G. Voelker, "Identifying suspicious urls: An application of large-scale online learning," in *Proceedings of the 26th Intl. Conf. on Machine Learning (ICML)*, 2009, pp. 681–688.

[19] McAfee, "McAfee TrustedSource web database reference guide," Mar. 2010.