Detecting Hidden Propagation Structure and Its Application to Analyzing Phishing

Yang Liu, Mingyan Liu
Electrical Engineering and Computer Science
University of Michigan, Ann Arbor
{youngliu, mingyan}@umich.edu

***Abstract*—In this paper we study the problem of how to detect and extract a particular type of propagation structure that arises in phishing activities. One of the most interesting phenomena induced by phishing is fast-flux, whereby a single malicious domain is mapped to a constantly changing IP address in order to evade capture and shut-down. This leads to malicious activities observed to be propagating through different networks, even though they originate from the same phishing campaign. To be able to detect and extract such a propagation is of significant importance as it can help us understand and analyze phishing activities. To achieve this goal, we propose a multi-layered propagation model, where layers correspond to different delay stages in the propagation and each is given by an adjacency matrix called the propagation matrix which models pairwise propagation relationships. A regression problem is then formulated to estimate this set of matrices so that the model prediction best fits the data; a Gibbs sampling based randomized algorithm is developed to efficiently find solutions with guaranteed performance. We evaluate our method using both simulation and Internet measurement data.**

***Index Terms*—Multi-layer propagation model, propagation detection, regression, phishing, measurement, network-level malicious activities**

I. Introduction

In many natural and engineered systems, risks and their manifestations propagate throughout the system. The mechanisms underlying such propagation differ from case to case, but it is often enabled by the inter-connectedness and interdependencies among the components of the system. Prime examples include for instance the spread of contagious diseases among a population [16], [20] and cascading failures in a power grid [17].

Understanding how risks propagate in these various settings is often of vital importance: it tells us how to design more resilient systems, e.g., by taking away (or strengthening) the most critical node [13]. Another example is [23] which showed that the extinction probability and propagation rate of epidemics over a network is closely related to the eigenvalue of the underlying topological structure. At the same time, when the underlying mechanism driving the propagation is unknown, e.g., how a particular disease is passed from one person to another, or how the failure of one switch on the grid triggers another, empirical observations on the temporal evolution of the outbreak (resp. failures) can often shed light on the unknown mechanism.

In this paper we study the problem of how to detect and extract a particular type of propagation that arises in phishing activities. One of the most interesting phenomena induced by phishing is fast-flux [18], whereby a single malicious domain is mapped to a constantly changing IP address in order to evade capture and shut-down. This leads to malicious activities observed to be propagating through different networks, even though they originate from the same phishing campaign. To be able to detect and extract such a propagation amounts to identifying the set of networks involved in the same phishing campaign, and is thus of significant importance as it can help us understand and analyze phishing activities.

Conceptually detecting propagation groups bears some resemblance to community detection [15], whose target is to find a subset within a network that shares some type of similarity. However, technically our task is very different from most of the existing community detection literature. This is because existing literature largely focuses on detecting communities in a static setting, e.g., a cluster within a network with static topology. On the other hand, propagation only arises in a dynamic setting, e.g., with data given as a set of temporal signals rather than a static graph. Thus we must be able to capture similarity over time and in a way that's consistent with the type of propagation. This is a much more challenging task.

Dynamic connectivity structure has been studied in a related domain, namely social networks, where different methods have been proposed to track information diffusion paths. For example, in [22] Rodriguez et al. proposed a method to uncover information diffusion patterns based on temporal dynamics. Later [26] generalized this methodology by considering heterogeneous link functions. However, all of the above depends heavily on the notion of events, requiring as input a set of event occurrences and timelines that can help pin down a set of propagation paths corresponding to the events. This makes the problem very different from ours as the security incidences usually remain unknown and in general no source information (e.g., attack sources, campaigns, causal effects, etc) is available.

This material is based on research sponsored by the Department of Homeland Security (DHS) Science and Technology Directorate, Homeland Security Advanced Research Projects Agency (HSARPA), Cyber Security Division (DHS S&T/HSARPA/CSD), BAA 11-02 via contract number HSHQDC-13-C-B0015.

To achieve our goal, we propose a multi-layered propagation model, where layers correspond to different delay stages in the propagation and each is captured by an adjacency matrix called the propagation matrix. A regression problem is then formulated to estimate this set of matrices so that the model prediction best fits the data. The computation of this regression problem is shown to be non-trivial due to its exponentially growing solution space w.r.t. the data size (the number of units in the network), which is large in our case (on the order of hundreds of thousands). To alleviate the computational complexity, we propose a Gibbs sampling based randomized algorithm to solve this problem with guaranteed convergence and ϵ-optimality. We demonstrate the effectiveness of the methodology by both simulation and using a set of empirical data.

While this method is more broadly applicable, to be concrete we will frame our discussion specifically within the context of phishing. To do so we will start with a description of our data, which is a set of well known host reputation blacklists (RBLs, summarized in Table I) collected over a period of 10 months (Jan - Oct 2013). These lists record, on a daily basis, host IP addresses seen engaged in some malicious activities – they are loosely broken down into three types: spam, phishing, and scanning attack. To capture how phishing campaigns propagate from one network to another, we aggregate the host information given in the RBLs at the prefix level, i.e., we record the daily fraction of IPs belonging to a prefix that get listed, referred to as an aggregate temporal signal. We then define a similarity measure between two prefixes' aggregate signals by correlating the two. Putting together all pairwise similarity measures leads to a similarity graph. By projecting the similarity graph onto its principle directions, evidence of propagation immediately emerges. We will show that our proposed propagation model and solution methodology is very effective in identifying and extracting these propagation groups.

Our major contributions are two-fold:

1) We propose a framework to uncover propagation patterns from a large number of un-labeled temporal signals.
2) To our best knowledge, this is the first systematic study towards revealing the propagation of malicious activities by utilizing Internet-scale measurement data.

The rest of the paper is organized as follows. We introduce our dataset in Section II and present empirical results on the similarity between networks' malicious activities and evidence of propagation in phishing campaigns. In Section III we present a regression model designed to extract propagation groups seen in the phishing data. The result is tested using both simulation and our RBL data in Section IV. Section V concludes our works.

II. The Dataset and Preliminaries

A. The RBL dataset

Our dataset consists of 11 IP address-based reputation blacklists over the 10-month period from January to October 2013. The sampling rate is once per day (i.e., the list content is refreshed on a daily basis). Table I summarizes these lists and the type of malicious activities they target: spam, phishing/malware, and active scanning. All combined this dataset includes 164 million unique IP addresses. Even though our later analysis will only focus on the phishing data, in this section we show features of all three types for contrast and to better highlight the propagation seen in the phishing data.

Type	Blacklist Name
Spam	CBL [2], SBL [10], SpamCop [8], WPBL [12], UCEPOTECT [11]
Phishing/Malware	SURBL [9], Phish Tank [6], hpHosts [4]
Active scanning	Darknet scanner list [], Dshield [3], OpenBL [5]

TABLE I
The RBL datasets

We first aggregate the individual blacklists in a union fashion along each malicious activities, i.e., an IP is listed on an union list for one type of activity on a given day as long as it shows up on at least one of the individual blacklists of that type on that day. This leads to three union lists, referred to as the Spam, Phishing, and Scanning lists, respectively.

These union lists remain at the IP level, i.e., they contain information on specific IP addresses, while we wish to capture activities at a network level to be able to discern propagation. For this reason we next aggregate the listed IP addresses at the routed prefix level (collected by all vantage points of Route Views [21] and RIPE [7] projects), by counting the number of listed IP addresses within each prefix. We thus obtain, for a given union list, a discrete-time *aggregate signal* for each prefix i; these are denoted by $r_i^{sp}(t)$, $r_i^{ph}(t)$, $r_i^{sc}(t)$, $t = 0, 1, 2, \cdots$, for signals obtained from the spam, phishing and scanning lists, respectively.

There are two types of aggregate one can define: the normalized version and the un-normalized version. For the normalized version, $r_i^*(t)$ is given by the fraction of the total number of IPs on the $*$-list and belonging to prefix i on day t over the total number of addresses within prefix i; here we use $*$ to denote any of the union lists. In the un-normalized version $r_i^*(t)$ is simply defined as the total number of IPs on the $*$-list and belonging to prefix i on day t. Note that when we examine prefixes of the same size, the two aggregate definitions are equivalent. When we examine prefixes of different sizes, the normalized version hides the actual number of malicious IPs so that a large-sized prefix does not overwhelm a small one when they have the same percentage of IPs listed. As prefixes are of different sizes in this study we will use the normalized version of the aggregate signals. In all, the RBL dataset represents 363,667 unique prefixes.

Over the 10-month period, a majority of the prefixes have very low presence on these RBLs (over 80% of prefixes have $< 2\%$ of their IPs listed on average), while a small number ($< 1\%$) has a very high number of malicious IPs. For this as well as computational reasons, throughout our analysis we will limit our attention to the top 5,000 most

significant (or malicious by this average measure) prefixes given a particular union list. Note that different union lists result in different sets of top 5,000 most malicious prefixes, e.g., those heaviest in scanning activities may not be the same as those in spamming, and so on.

B. Similarity measure and similarity graphs

Given two aggregate signals $r_i^*(t)$ and $r_j^*(t)$ for two prefixes i, j, we next define a similarity measure and examine to what extent these dynamic network-level malicious behaviors are similar to each other. We then project the similarity matrix onto a 2D graph which gives us direct evidence of propagation effects.

Let $\mathbf{r}_i^*$ and $\mathbf{r}_j^*$ be the vector form of $r_i^*(t)$ and $r_j^*(t)$, $t = 1, \cdots, \tau$, respectively for some horizon τ. Then the *temporal similarity* between these two vectors can be measured by the following:

$$S_{i,j}^* = \frac{2(\mathbf{r}_i^*)^T \cdot \mathbf{r}_j^*}{|\mathbf{r}_i^*|^2 + |\mathbf{r}_j^*|^2}, \ \forall i \neq j \ , \tag{1}$$

where T in the superscript denotes transpose. The above is essentially a correlation measure, so its meaning is straightforward: it captures how similar in shape these two vectors are. Given N prefixes, the collection of N^2 similarity values can be represented in a *similarity matrix* $S^* = [S_{i,j}^*]$. It is easy to see that S^* may be interpreted as a weighted adjacency matrix for an underlying similarity graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: V is the set of N prefixes, $\mathcal{E}$ the set of weighted edges, with weights $S_{i,j}^*$ representing the closeness/ similarity between two connected prefixes $i, j \in \mathcal{V}$.

We next visualize these similarity graphs on a 2D plane, on which each point represents a prefix, and the pair-wise Euclidean distance between every two points is approximately inversely proportional to their edge weights. Thus the closer two points are, the more similar the corresponding prefixes in their aggregate signals. The approximation is due to the fact that in computing the locations of these prefixes whose distances satisfy the set of pairwise similarity measures, the true answer generally lies in a higher dimensional space, in which case the 2D depiction is the projection of the true solution to the 2D plane, or a least-squares approximation.

We plot these similarity graphs derived from the three union lists for the month of October 2013 (i.e., from matrices S^{sp}, S^{ph}, and S^{sc}, respectively). These are shown in Figure 1. In inspecting Figures 1(a)-1(c), we note very different features. In particular, the phishing graph shows distinct curves/lines. This indicates a type of "continuity" in similarity, i.e., successive neighbors are very close to each other but they collectively span over a much larger distance. The most direct explanation for this phenomenon is that the sequence of prefixes share similar aggregate signals but with a progressive phase shift; equivalently one may view all prefixes as capturing the same signal propagating through them.

To verify this explanation, consider a set of signals shown in Figure 2(a), where each is a delayed version of the previous one. When we plot the 2D projection of the resulting similarity matrix for this set of signals, indeed a line is observed as shown in Figure 2(b). We further verify this explanation by extracting a set of prefixes from the top two curves of the phishing graph; their aggregate signals are shown in Figure 2(c)−2(d). These show quite clearly the propagation effect illustrated above.

The reason behind the observed propagation effect is the fast-flux phenomenon [18] mentioned earlier, whereby a single malicious domain is mapped to a constantly changing IP address. This leads to a single malicious event propagating through different prefixes over time with the result that two successive prefixes exhibit high similarity (separated by coordinated shifts) in their dynamic behavior.

For the rest of the paper we will solely focus on the propagation phenomenon observed in the phishing similarity graph shown in Figure 1(b), and will try to address the question whether we can develop an algorithm to extract these propagation groups from such a graph. The reason why we are interested in this has to do with forensics: if we can quickly extract the set of networks involved in a propagation group, we can then find out the IP addresses within those networks that are responsible for the phishing activity and during what time period, and verify (by checking the URLs/websites associated with those IP addresses) whether they originate from the same campaign. This allows us to further analyze how a phishing campaign chooses the sequence of IP addresses and how it migrates from one network to another.

III. Detecting and Extracting propagation groups

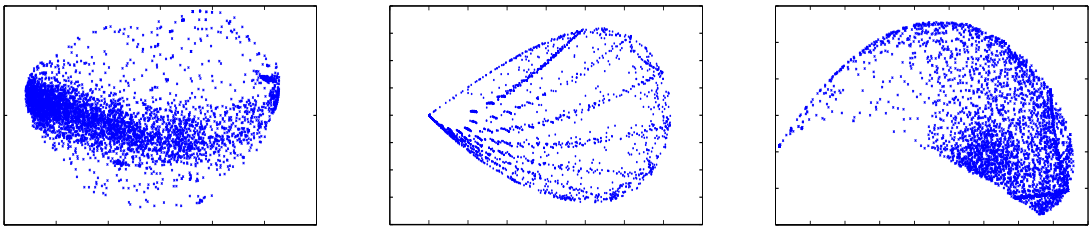
With empirical evidence of propagation shown in the previous section, we next detail a multi-layer propagation model that aims at capturing the underlying propagation mechanism. We then formulate a regression problem towards finding propagation groups from empirically measured temporal signals without knowledge of possible triggering incidents.

A. A multi-layered propagation model

Denote by N the number of network units (prefixes in our context) under consideration. The duration of observation is set to be τ days, i.e., we have historical data for time $t = 1, 2, ..., \tau$ for each unit. The observed maliciousness of prefix i at time t, i.e., the value of its aggregate signal at time t, is $r_i(t)$.

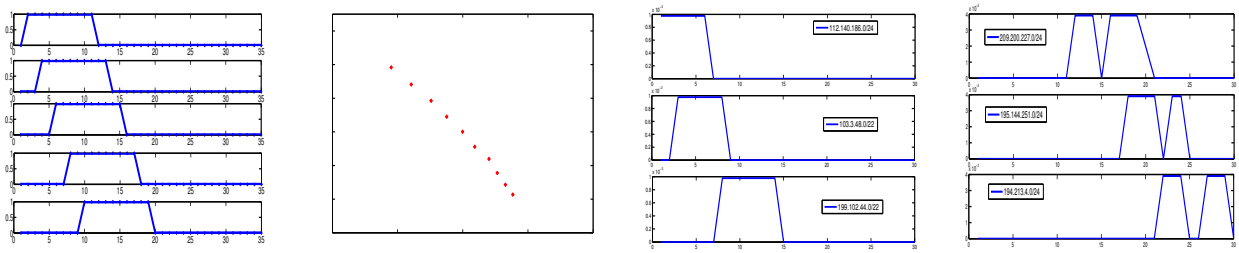
Consider a hidden propagation graph $\mathcal{G}$ connecting the networks/prefixes; each prefix i is connected to a set of neighbors taking into account time delay: if what happens in network j reaches network i within k days, then we say network j is in i's k-neighborhood on the graph $\mathcal{G}$. Denote this neighborhood by $\mathcal{N}_i^k$. We will model the propagation effect among the networks as follows:

$$\hat{r}_i(t+1) = \beta \cdot \sum_{k=1}^{d} \sum_{j \in \mathcal{N}_i^k} r_j(t+1-k) + (\lambda_i - \mu_i) r_i(t) \ , \tag{2}$$



(a) Spam (b) Phishing (c) Scan

Fig. 1. 2D visualization of similarity graphs : different malicious activities. Our focus is on phishing, with the other two illustrated to provide contrast.



(a) Simulated signals. (b) Similarity graph. (c) Phishing : Example 1 (d) Phishing : Example 2

Fig. 2. Illustration of propagation. (a) shows the simulated signals propagating through the group. The resulting similarity graph is given in (b). Further verification is done by extracting two sets of propagation groups from the top two curves in the phishing similarity graph, shown in (c) and (d).

where $\hat{r}_i(t+1)$ is the estimated signal value (or the regression outcome) for network i on day $t+1$; we will try to match this to the observed value $r_i(t+1)$ as closely as possible. The interpretation of this model is that what is seen for network i on day $t+1$ is a combination of what was seen within its k-neighborhood within the previous k days, and its own condition the day before. This would capture the following: (1) an IP address in network j used for phishing migrated to network i on day $k = 1, \cdots, d$, and (2) an IP address in network i used for phishing the previous day continues to be used for phishing on the current day.

In Eqn (2), β is an attenuation parameter discounting the propagation effect as it reaches i from its neighbors; λ_i is the arrival rate of external risk (or newly selected hosts to run phishing sites in our context); μ_i is the extinction rate of a prefix (which counts for phishing hosts migrating out of the network or cleaning up by the local host/system administrator); and d is the propagation horizon we consider.

We discuss briefly various ways in which the above model can be further enhanced. For instance, all the parameters can potentially be made time-varying, and the attenuation β can depend on the source and destination networks i and j. Some thoughts also need to go into selecting the right value for d: this should ideally be consistent with the typical duration of a phishing campaign. However, very little is known about this quantity in the literature. From a computational point of view a small d is preferred as it decreases complexity. It is also preferred because it reduces the chance of mistaking coincidence for true synchronized behavior. For these reasons in our numerical experiment we will keep this value relatively small ($d = 7$). As we will see a consequence of this is that a single phishing campaign may be identified and extracted as multiple smaller propagation groups. This is not a serious drawback as smaller groups can be easy assembled afterwards.

The above propagation model can be written in matrix form:

$$\hat{\mathbf{r}}(t+1) = \beta \cdot \sum_{k=1}^{d} A_k \cdot \mathbf{r}(t+1-k) + (\Lambda - \Pi) \cdot \mathbf{r}(t) \ , \quad (3)$$

where Λ, Π are diagonal matrices with $\Lambda = \text{diag}(\lambda_i)$ and $\Pi = \text{diag}(\mu_i)$. $A_1, ..., A_d$ are referred to as *propagation matrices* corresponding to different delay (up to d days), respectively. Specifically, if $A_k(i,j) = 1$ we say there is direct k-day propagation from prefix j to prefix i. The propagation matrices are unweighted adjacency matrices, i.e., $A(i,j) \in \{0,1\}$.

For the rest of the analysis we will focus on a simplified model: no attenuation ($\beta = 1$) and steady state operation ($\lambda_i = \mu_i$). The recursive equation then reduces to the following,

$$\hat{\mathbf{r}}(t+1) = \sum_{k=1}^{d} A_k \cdot \mathbf{r}(t+1-k) \ . \quad (4)$$

Similar to what is commonly done in community detection, we will try to identify the propagation matrices by finding the best fit for the set of original signals $\{\mathbf{r}_i\}_i$. That is, we will find the set of $\{A_k\}$ that minimizes certain loss function with input being $(\mathbf{r}, \hat{\mathbf{r}})$:

$$\min_{\{A_k\}} \mathcal{L}(\mathbf{r}, \hat{\mathbf{r}}) \ ,$$

subject to certain constraints on $\{A_k\}$. For example, taking $\mathcal{L}(\cdot)$ to be the squared error norm results in the following optimization problem:

$$\begin{array}{ll}\min_{\{A_k\}} & \sum_{t=1}^{\tau} ||\mathbf{r}(t) - \hat{\mathbf{r}}(t)||_2^2 \\ \text{s.t.} & A_k(i,j) \in \{0,1\}, k = 1, ..., d, i, j \in \mathcal{U}. \\ & \hat{\mathbf{r}}(t) = \sum_{k=1}^{d} A_k \cdot \mathbf{r}(t-k). \\ & \sum_{k=1}^{d} A_k(i,j) \leq 1, \forall (i,j) \in \mathcal{U}^2 \ , \end{array}$$

where the last constraint stipulates that propagation between two networks happen no more than once within the period d.

B. Additional modeling issues

There are a few more modeling issues worth discussing.

1) Suppose there is a propagation chain $i \to j \to l$. Then the fitting error would be the same for l whether we have a directed edge from $i \to l$ in matrix A_2 or one from $i \to j$ and $j \to l$ in matrix A_1. The optimization problem can be modified (with additional constraints) to induce either type of solutions. In the present study we will choose to prefer solutions of the first type, i.e., direct (even if delayed) propagation.
2) Without constraining the number of edges in the propagation matrix, the optimization problem tends to add more edges than desired. This is a well-known problem called over-fitting.

To address the above issues, we consider the following modification to the original regression problem:

$$\begin{array}{ll}\min_{\{A_k\}} & \sum_{t=1}^{\tau} ||\mathbf{r}(t) - \hat{\mathbf{r}}(t)||_2^2 + \sum_{k=1}^{d} \gamma^{-k} \cdot ||A_k||_1 \\ \text{s.t.} & A_k(i,j) \in \{0,1\}, k = 1, ..., d, i, j \in \mathcal{U}. \\ & \hat{\mathbf{r}}(t) = \sum_{k=1}^{d} A_k \cdot \mathbf{r}(t-k). \\ & \sum_{k=1}^{d} A_k(i,j) \leq 1, \forall (i,j) \in \mathcal{U}^2. \end{array}$$

$\sum_{k=1}^{d} \gamma^{-k} \cdot ||A_k||_1$ is a regulation term and $\gamma \in (0,1)$ is the regulation parameter. It serves as a penalty to induce fewer edges in the propagation matrices. Moreover since γ^{-k} is increasing in k, we punish more an edge in a high order propagation matrix (longer delay), reflecting less confidence in establishing a link. We refer to this problem as **(PR)**.

Proposition III.1. *If the propagation model is accurate, then the optimal solution returned by* **(PR)** *will not mistake a longer path (say $i \to j \to l$ with delays d_{ij} and d_{jl}) for a shorter one ($i \to l$ with delay $d_{il} = d_{ij} + d_{jl}$).*

Proof. We prove this by contradiction. Suppose the true propagation path contains the segment $i \to j \to l$ with delays d_{ij} and d_{jl} (i.e., there is an edge $i \to j$ in $A_{d_{ij}}$ and edge $j \to l$ in $A_{d_{jl}}$), but the optimal solution returns the edge $i \to l$ in $A_{d_{ij}}$ with $d_{il} = d_{ij} + d_{jl}$. We first establish that the edge $i \to j$ must also exist in the optimal solution. This is because the existence of edge $i \to l$ means that its inclusion carries more reduction in the fitting error than the regulation penalty it induces. It follows that the edge $i \to j$ must also exist since its inclusion carries the same amount of reduction in fitting error (since the model is assumed accurate) while resulting in less regulation penalty.

Having established that $i \to j$ exists, we now add edge $j \to l$ to matrix $A_{d_{jl}}$ and remove the edge $i \to l$ from $A_{d_{il}}$. It is easy to verify the first fitting term, i.e., $\sum_{t=1}^{\tau} ||\mathbf{r}(t) - \sum_{k=1}^{d} A_k \cdot \mathbf{r}(t-k)||_2^2$, stays unchanged after this addition. However, the regulation term undergoes the following change:

$$\gamma^{-d_{il}} \cdot ||e_i \cdot e_l^T||_1 \Rightarrow \gamma^{-d_{jl}} \cdot ||e_j \cdot e_l^T||_1 \ . \tag{5}$$

Since $d_{il} > d_{jl}$, $\gamma < 1$, we have

$$\gamma^{-d_{il}} \cdot ||e_i \cdot e_l^T||_1 > \gamma^{-d_{jl}} \cdot ||e_j \cdot e_l^T||_1 \ .$$

This means we have found a better solution, which contradicts the optimality assumption. The same argument can be repeatedly applied to any segment of a propagation path. □

C. A Gibbs sampling based fast recovery

The problem presented above is hard to solve (solution space is a set of integer values) and solving the problem directly is computationally prohibitive. To see this note that we have $d \cdot N^2$ variables, and for a generic integer program the solution complexity is exponential in this number. We thus propose the following randomized solution approach which can guarantee ϵ-optimality, $\forall \epsilon > 0$. Consider adding an edge from network j to i in the propagation matrix A_k, i.e.,

$$A_k \Leftarrow A_k + e_i \cdot e_j^T \ . \tag{6}$$

We would like to find the resulting change in the objective function value due to this addition. Denote by

$$\begin{aligned} &\mathcal{J}^*(A_1, A_2, \cdots, A_d) \\ &= \sum_{t=1}^{T} ||\mathbf{r}(t) - \sum_{k=1}^{d} A_k \cdot \mathbf{r}(t-k)||_2^2 \\ &\quad + \sum_{k=1}^{d} \gamma^{-k} \cdot ||A_k||_1 \ . \end{aligned} \tag{7}$$

Consider $\mathcal{J}^*(A_{-k}, A_k + e_i \cdot e_j^T)$ and inspect each term in the summation of $\mathcal{J}^*$. The change to the regulation term is easy to obtain and given by:

$$\gamma^{-k} \cdot ||A_k + e_i \cdot e_j^T||_1 - \gamma^{-k} \cdot ||A_k||_1 = \gamma^{-k} \ . \tag{8}$$

Consider now the fitting term, which can be decomposed into the following:

$$\begin{aligned} &||\mathbf{r}(t) - \sum_{k=1}^{d} A_k \cdot \mathbf{r}(t-k)||_2^2 \\ &= \underbrace{\mathbf{r}^T(t) \cdot \mathbf{r}(t)}_{\textbf{Term 1}} \underbrace{-2 \cdot \mathbf{r}^T(t) \cdot \sum_{k=1}^{d} A_k \cdot \mathbf{r}(t-k)}_{\textbf{Term 2}} \\ &\quad + \underbrace{\left[\sum_{k=1}^{d} A_k \cdot \mathbf{r}(t-k)\right]^T \cdot \left[\sum_{k=1}^{d} A_k \cdot \mathbf{r}(t-k)\right]}_{\textbf{Term 3}} \ . \end{aligned} \tag{9}$$

First note **Term 1** is independent of the choice of $\{\mathcal{A}\}$; thus we will only focus on **Term 2** and **Term 3**. Replacing A_k with $A_k + e_i \cdot e_j^T$ we record the changes as follows:

$$\begin{aligned}\Delta(\textbf{Term 2}) &= -2 \cdot \mathbf{r}^T(t) \cdot (e_i \cdot e_j^T) \cdot \mathbf{r}(t-k) \\ &= -2 \cdot r_i(t) \cdot r_j(t-k) \ , \end{aligned} \tag{10}$$

$$\begin{aligned}\Delta(\textbf{Term 3}) &= 2 \cdot \mathbf{r}^T(t-k) \cdot (e_j \cdot e_i^T) \cdot \sum_{k=1}^{d} A_k \cdot \mathbf{r}(t-k) \\ &+ \left[e_i \cdot e_j^T \cdot \mathbf{r}(t-k)\right]^T \cdot \left[e_i \cdot e_j^T \cdot \mathbf{r}(t-k)\right] \ . \end{aligned} \tag{11}$$

Simplifying further we have

$$\begin{aligned}&2 \cdot \mathbf{r}^T(t-k) \cdot (e_j \cdot e_i^T) \cdot \sum_{k=1}^{d} A_k \cdot \mathbf{r}(t-k) \\ &= 2 \cdot r_j(t-k) \cdot \sum_{k=1}^{d} A_k(i,:) \cdot \mathbf{r}(t-k) \ , \end{aligned} \tag{12}$$

and

$$\left[e_i \cdot e_j^T \cdot \mathbf{r}(t-k)\right]^T \cdot \left[e_i \cdot e_j^T \cdot \mathbf{r}(t-k)\right] = r_j^2(t-k) \ , \tag{13}$$

where we have used $A_k(i,:)$ to denote the ith column of matrix A_k. Summing up all changes and define the potential function for edge $j \to i$ on A_k as

$$\begin{aligned}\Delta_k(i,j) = \sum_{t=1}^{T} \Bigg\{ &-2 \cdot r_i(t) \cdot r_j(t-k) + 2 \cdot r_j(t-k) \\ &\cdot \sum_{k=1}^{d} A_k(i,:) \cdot \mathbf{r}(t-k) + r_j^2(t-k) \Bigg\} \\ &+ \gamma^{-k} \ . \end{aligned} \tag{14}$$

This change in the objective function value consists of four parts. The last part counts as regulation while the first three come from fitting.

Based on the above results we propose the following randomized algorithm which targets an ϵ-optimal solution. The algorithm works in an iterative fashion and starts with an initial set of propagation matrices $A_1, A_2, \cdots A_d$. At each step, we first evaluate the objective function value given the current set of propagation matrices; denote that by U. Then a pair of nodes (i,j) is randomly selected among all node pairs. Remove current edges $j \to i$ from all A_k's if they exist. Then use the resulting set $\{A_k\}_{k=1}^d$ to evaluate $\Delta_k(i,j)$, i.e., the difference its addition would make to the current objective function. Then for $k = 1, 2, \cdots, d$, add this edge to matrix A_k with probability (using a d-sided coin)

$$\begin{aligned}P_{j\to i}(A_k) &= \frac{e^{-\frac{U+\Delta_k(i,j)}{\sigma^2}}}{\sum_{m=1}^{d} e^{-\frac{U+\Delta_m(i,j)}{\sigma^2}} + e^{-\frac{U}{\sigma^2}}} \\ &= \frac{e^{-\frac{\Delta_k(i,j)}{\sigma^2}}}{\sum_{m=1}^{d} e^{-\frac{\Delta_m(i,j)}{\sigma^2}} + 1} \ . \end{aligned} \tag{15}$$

The probability of adding no edge is given as

$$P_{j\to i}(\varnothing) = \frac{1}{\sum_{m=1}^{d} e^{-\frac{\Delta_m(i,j)}{\sigma^2}} + 1} \ . \tag{16}$$

The process continues till it converges, defined as improvement to the objective value in successive rounds falling below a threshold for a long enough period.

Theorem 1. *The randomized algorithm is ϵ-optimal asymptotically.*

Proof. The proof follows from standard techniques of Gibbs sampling and is thus omitted for brevity. □

Besides guaranteed performance, this algorithm is also computationally light. Denote the number of steps required to reach convergence by C. Then the computation complexity is on the order of $\mathcal{O}(C \cdot N^2)$, which is polynomial instead of exponential in N.

IV. Validation

We evaluate our algorithm using both numerical simulation and by applying it to the RBL phishing data shown earlier. For simulation, we insert a propagation group of size k into a network of size N. For members outside this group their signals are sequences of IID values drawn from a uniform distribution. For members within the propagation group their signals are given by those shown in Figure 2(a), a 5-day signal with a progressive time shift. The simulation is run long enough to allow the above signal to fully propagate through a k-node group, i.e., at least $k+5$ days in simulation time.

$N \backslash k$	5	10	20	50
50	100%	75%	73.68%	(100%)
100	100%	77.78%	63.16%	56.2%
200	100%	66.67%	57.89%	57.14%
300	100%	66.67%	68.42%	55.1%
1,000	100%	55%	65.10%	36%

TABLE II
Detection accuracy. Detection is very accurate when k is small; it degrades as k increase. However, in all the detection rates remain above 55%.

The detection performance is summarized in Table II (Note that the "(100%)" entry is not particularly meaningful: in this case the entire set of nodes belongs to the propagation group and the algorithm is able to recover the whole group but this is in the absence of other signals in the system). A specific example is shown in Figure 4(a), where the blue dots denote the propagation group and is precisely taken out by our algorithm. The average performance summarized in Table II suggests that up to groups of size 5, our algorithm is very accurate. The performance degrades as the group becomes larger, which is to be expected as longer propagation paths make it harder to extract all members correctly in sequence.

The convergence speed of our algorithm varies depending on the randomly generated tests. However, most of them converge quickly within 100 rounds of simulation cycles. Some examples are plotted in Figure 3.

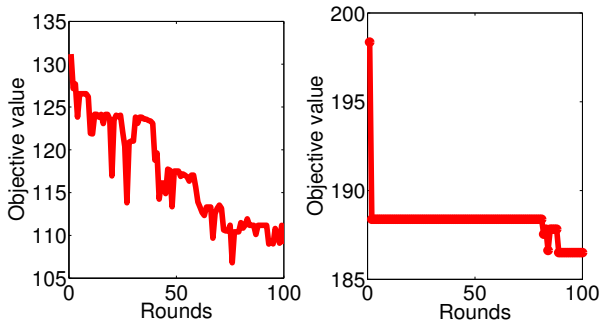
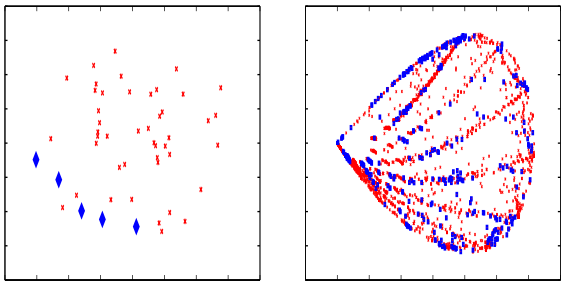


Fig. 3. Examples of convergence



(a) Simulation. (b) Phishing.

Fig. 4. Validation results. a). A propagation group of 5 nodes was precisely extracted (shown in blue). b). Selected propagation groups are superimposed (in blue) on the original 2D visualization. Key parameters used: Size of the system $= 5,000$, $\tau = 30$, $d = 7$, $\gamma = 0.4$.

We then apply the method to the phishing data shown earlier. Figure 4(b) shows the extraction result where the original graph (in red) is superimposed with blue dots indicating groups identified and extracted by our algorithm. Here we have set $d = 7$, i.e., we consider propagation effects up to a week's delay. As expected, since we only allow small group sizes, our algorithm returned a large number of groups with different maximum delays. For simplicity of presentation, we have superimposed a subset of these groups on the same graph while mentioning that each individual group is always closely located and along the same linear group seen in the original graph. Table III shows the distribution of the sizes of the groups extracted (the maximum size is 8 given the largest allowed delay is 7).

Group size	2	3	4	5	6	7	8
Groups	1,920	468	130	38	15	0	0

TABLE III
IDENTIFIED GROUPS

V. CONCLUSION

In this paper we presented a method aimed at identifying and extracting groups involved in the propagation of phishing activities. We proposed a multi-layered propagation model, where layers correspond to different delay stages in the propagation and formulated a regression problem to find the best match between the model prediction and the data. Evaluation using simulation and Internet measurement data showed that our method can be very effective.

REFERENCES

[1] Barracuda Reputation Blocklist. http://www.barracudacentral.org/.
[2] Composite Blocking List. http://cbl.abuseat.org/.
[3] DShield. http://www.dshield.org/.
[4] hpHosts for your pretection. http://hosts-file.net/.
[5] OpenBL. http://www.openbl.org/.
[6] PhishTank. http://www.phishtank.com/.
[7] RIPE Routing Information Service (RIS) Raw data Project. http://www.ripe.net/data-tools/stats/ris/ris-raw-data.
[8] SpamCop Blocking List. http://www.spamcop.net/.
[9] SURBL: URL REPUTATION DATA. http://www.surbl.org/.
[10] The SPAMHAUS project: SBL, XBL, PBL, ZEN Lists. http://www.spamhaus.org/.
[11] UCEPROTECTOR Network. http://www.uceprotect.net/.
[12] WPBL: Weighted Private Block List. http://www.wpbl.info/.
[13] Pin-Yu Chen and A.O. Hero. Node removal vulnerability of the largest component of a network. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pages 587–590, Dec 2013.
[14] Amogh Dhamdhere and Constantine Dovrolis. Ten years in the evolution of the internet ecosystem. In *Proceedings of ACM IMC*, pages 183–196, New York, NY, USA, 2008. ACM.
[15] Nan Du, Bin Wu, Xin Pei, Bai Wang, and Liutong Xu. Community detection in large-scale social networks. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007*, WebKDD/SNA-KDD '07, pages 16–25, New York, NY, USA, 2007. ACM.
[16] A. Ganesh, L. Massoulie, and D. Towsley. The effect of network topology on the spread of epidemics. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 1455–1466 vol. 2, March 2005.
[17] Paul Hines, Karthikeyan Balasubramaniam, and Eduardo Cotilla Sanchez. Cascading failures in power grids. *Potentials, IEEE*, 28(5):24–30, 2009.
[18] Thorsten Holz, Christian Gorecki, Konrad Rieck, and Felix Freiling. Measuring and detecting fast-flux service networks. In *In Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*. ISOC, 2008.
[19] Chris Kanich, Christian Krebich, Kirill Levchenko, Brandon Enright, Geoffrey Voelker, and Vern Paxon. Spamalytics: An empirical analysis of spam marketing conversion. In *Proceedings of the 15th ACM conference on Computer and communications*, pages 3–14. ACM, 2008.
[20] Marc Lelarge. Economics of malware: Epidemic risks model, network externalities and incentives. In *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, pages 1353–1360. IEEE, 2009.
[21] University of Oregon. Route Views Project. http://www.routeviews.org/.
[22] Manuel Gomez Rodriguez, David Balduzzi, and Bernhard Schölkopf. Uncovering the temporal dynamics of diffusion networks. *arXiv preprint arXiv:1105.0697*, 2011.
[23] Yang Wang, D. Chakrabarti, Chenxi Wang, and C. Faloutsos. Epidemic spreading in real networks: an eigenvalue viewpoint. In *Reliable Distributed Systems, 2003. Proceedings. 22nd International Symposium on*, pages 25–34, Oct 2003.
[24] Yinglian Xie, Fang Yu, Kannan Achan, Eliot Gillum, Moises Goldszmidt, and Ted Wobber. How dynamic are ip addresses? In *Proceedings of SIGCOMM*, pages 301–312, New York, NY, USA, 2007. ACM.
[25] Jing Zhang, Ari Chivukula, Michael Bailey, Manish Karir, and Mingyan Liu. Characterization of Blacklists and Tainted Network Traffic. In *Proceedings of PAM*, Hong Kong, March 2013.
[26] Ke Zhou, Hongyuan Zha, and Le Song. Learning triggering kernels for multi-dimensional hawkes processes. In Sanjoy Dasgupta and David Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 1301–1309. JMLR Workshop and Conference Proceedings, May 2013.