STOP: Socio-Temporal Opportunistic Patching of Short Range Mobile Malware

John Tang
University of Cambridge

Hyoungshick Kim
University of British Columbia

Cecilia Mascolo
University of Cambridge

Mirco Musolesi
University of Birmingham

***Abstract*—Mobile phones are integral to everyday life with emails, social networking, online banking and other applications; however, the wealth of private information accessible increases economic incentives for attackers. Compared with fixed networks, mobile malware can replicate through both long range messaging and short range radio technologies; the former can be filtered by the network operator but determining the best method of containing short range malware is an open problem. While global software updates are sometimes possible, they are often not practical. An alternative and more efficient strategy is to distribute the patch to the key nodes so that they can opportunistically disseminate it to the rest of the network via short range encounters; but how can these key nodes be identified in a highly dynamic network topology?**

In this paper, we address these questions by presenting Socio-Temporal Opportunistic Patching (STOP), a two-tier predictive mobile malware containment system: devices collect co-location data in a decentralized manner and report to a central server which processes and targets delivery of hot fixes to a small subset of k devices at runtime; in turn mobile devices spread the patch opportunistically. The STOP system is underpinned by a recent theoretical framework for analysing dynamic networks that takes into account temporal information of links. Using empirical contact traces, we find firstly, the top-k ranking temporal centrality nodes are highly correlated with past time windows; and secondly, simple prediction functions can be designed to select the set of top-k nodes that are optimal for patch spreading.

I. Introduction

Smartphones have become increasingly powerful and their ubiquity and convenience has increased our reliance on them for both personal use (e.g., online social networks, location aware applications, mobile banking, NFC payment systems) and business applications (e.g., corporate emails, business calendar, client contact books and authentication to corporate network). With increasing economic incentives, in recent years we have seen a steady rise in mobile malware [13], [14], with exploits including remote access to devices [2], mobile banking attacks [4] and eavesdropping of calls [3].

Recently, approaches have been developed to reduce the threats of such malicious or poorly written programs [9], [15]; however, such techniques are mainly used to detect unauthorized access to information. Unlike these approaches, the solution presented here focusses on an effective security patch dissemination system to mitigate the spread of a mobile worm. Though prevention is very important, there are inherent detection limitations [7], hence, actively disseminating patches may be necessary.

This requirement is exacerbated by more diverse channels of propagation rendering mobile malware more sophisticated than desktop viruses; mobile malware can potentially exploit both long range (SMS, MMS, email etc.) and short range (Bluetooth, WiFi, ZigBee etc.) channels. Although a mobile network operator can potentially filter long range messages, this is not the case for short range propagation. The hazards of short range mobile worms were first highlighted when the first Bluetooth worm, *cabir* [1], broke out of a research lab by replicating to researcher's own devices; this prompted the need for a radio-shielded room for future research [12]. More recently, Rieback et al. demonstrated the feasibility of a RFID based worm propagation [17]. STOP addresses the important issue of containing short range transmission of mobile worms, by utilizing the same propagation channel which facilitates quick spreading via social contacts.

A. Motivation

There are several reasons why naively sending security patches directly to every device is not efficient in cellular environments. Firstly, the *cost* in mobile data service plans is not widely in favor of end users hence they may resist to receive security patches via 2G/3G networks if they do not subscribe to unlimited wireless data access plans; secondly, many mobile devices, such as tablets, do not have 2G/3G radios and hence rely on Bluetooth or WiFi to receive data; thirdly, there are inherent *restrictions in bandwidth* due to the current cellular infrastructure. In fact, it has been shown that the mobile infrastructure for the entire United States can be congested by only half-a-million messages per second [10]. Finally, *service coverage* is not guaranteed in certain areas (e.g., rural or underground metro systems).

We envisage that in scenarios where we may not be able to solely rely on the mobile network operator to deliver the patch, STOP can provide an alternative or complementary method for patch distribution. STOP uses both *social* and *temporal* information to target a small subset of devices to receive a patch and allow opportunistic contacts to spread the patch between devices.

Such proximity and social network based patch delivery mechanism have been introduced recently [23], [19], however, they show several limitations including:

- neglect periodic or regular human activity in a short-time scale [23];

978-1-4673-1239-4/12$31.00 ©2012 IEEE

- require that mobile devices perform non-trivial computation of containment strategies locally [23]; and
- are based on the assumption that global knowledge of all past *and* future contacts is available [19].

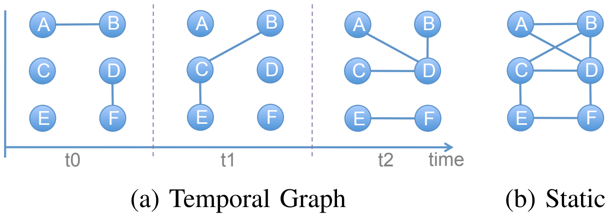
B. Contributions

We address these three shortcomings by introducing STOP (Socio-Temporal aware Opportunistic Patching system), a solution for malware containment that takes into account temporal contact patterns of real social behavior, offloads non-trivial computation to the mobile network operator, and predicts the best set of devices to start spreading a security patch[1]. STOP is composed of a two-tier model: mobile devices scan nearby devices using Bluetooth, Zigbee, NFC or other future short range radio technologies and upload these sightings periodically via WiFi, 3G or desktop sync. A *patch service* hosted by the mobile network operator is responsible for collecting the traces of *mobile services* and distributing the latest security patch to a key subset of central devices that then will spread the patch to neighbouring devices opportunistically via short range radio.

But *can we predict important nodes which can quickly spread a patch in order to stop malware dissemination?* To answer this question, we first have to take into account the rich temporal variation in the network of device contacts. Then, we need to evaluate the hypothesis that a currently important node is also likely to be important in the future (we verify this conjecture experimentally in Section III-A2). More specifically, firstly, we use techniques founded on time-varying social network analysis, namely *temporal graphs* to model the underlying contact process and *temporal closeness centrality* as a measure of node importance. Temporal closeness is a measure of a node's ability to spread information quickly to the most number of nodes [19]. Secondly, the definition of effective prediction functions requires the specification of a model which is able to capture the correlation of a node's importance in terms of information propagation with respect to the past history of the evolution of the contact network.

Our contributions can be summarized as follows:

- We define the *top-k set membership prediction problem* which is used to study the selection of a subset of nodes from which to start spreading a patch (see Section III-A). The goal of this *prediction model* is to forecast if a certain node can spread a message quickly and to the most number of devices in the future (more precisely, a member of the set of the top-k central nodes) based on the previous observations of the network evolution.
- By exploiting empirical datasets, we observe that the list of the top-k important devices shows a high correlation over time: linear time-complexity algorithms can be devised to predict the best devices to start spreading a patch (see Sections III-B & III-C).
- We evaluate node selection based on temporal closeness centrality against those founded on static closeness and random selection; we demonstrate that the most effective devices for spreading a patch can be identified through temporal centrality using a simple logarithmic weighted prediction function (see Section IV).



(a) Temporal Graph (b) Static

Figure 1. Example (a) temporal graph and corresponding (b) static graph.

Before we proceed, the next section shall motivate the need for temporal graph analysis and introduce the temporal centrality measure and empirical datasets used in this paper.

II. Preliminaries

A. Temporal Graph Model

Temporal graphs have recently been proposed [20] to model real time-varying networks, with the intuition that the behavior of dynamic networks can be more accurately captured by a sequence of *snapshots* of the network topology as it changes over time instead of using a representation whereby all the contacts are aggregated into a single static graph. For example, consider a temporal graph (Figure 1 (a)), consisting of 3 time windows and the corresponding aggregated static graph (Figure 1 (b)). The shortest path from node A to node F has 2 hops via (A, D, F); however, if we take into account the actual time order of contacts (Left of Figure 1), we notice that the first edge AD at time t_2 will never precede edge DF at time t_0; in fact the real temporal shortest path is 4-hops (A, B, C, E, F) (AB at time t_0, BCE at time t_1 and EF at time t_2) taking 3 time windows. Since static analysis ignores time ordering of contacts, static shortest paths *overestimate* the available links and *underestimate* the actual shortest path length; STOP takes advantage of this more realistic time-varying graph model. Also note that the high density of links within the static graph contributes to problems discriminating between important nodes when calculating static centrality [11]. Further, since closeness centrality is based on shortest paths, we use a *temporal closeness centrality* which takes advantage of this temporal information [19].

To take into account both uni- and bi-directional short range radio technologies, we define a *directed* temporal graph, which can be thought of as an ordered sequence of directed graphs[2].

[1] Such a security patch could be a malware signature or binary code to fix a known vulnerability.

[2] This does not lose generality in terms of bi-directional communication as transmissions can still be reciprocated during the same encounter.

More formally, given a real-world contact trace starting at t_{min} and ending at t_{max}, the directed temporal graph $\mathcal{G}^w(t_{min}, t_{max})$ is defined as the ordered sequence of graphs $(G_0,\ G_2, \ldots, G_{\tau-1})$ where $\tau = ((t_{max} - t_{min})/w) = |\mathcal{G}^w(t_{min}, t_{max})|$ is the number of graphs in the sequence and w is the size of each time window expressed in some time units (e.g., seconds or hours). There exists a directed link from i to j in G_t if there is a contact from i to j during the time interval $[(t_{min} + (w \times t)), (t_{min} + (w \times (t+1)))]$. All windows have the same set of nodes V.

From this, a *temporal path* starting at i and finishing at j can be defined over $\mathcal{G}^w(t_{min}, t_{max})$ as a sequence of η hops via a distinct node $n_\eta^{W_\eta}$ at time window W_η:

$$p_{ij}^h = (n_1^{W_1}, \ldots, n_\eta^{W_\eta}) \tag{1}$$

where $i = n_1$, $j = n_\eta$, node n_η is passed a message at time window $W_\eta \geq W_{\eta-1}$, $0 \leq W_\eta < \tau$ and h is the maximum number of hops through which a message is replicated within the same window. Subsequent definitions implicitly set $h = 1$, since higher values of h lead to similar performance of the containment schemes. We call Q_{ij} the set of all temporal paths between nodes i and j. If a temporal path from i to j does not exist, we set the distance $l_{ij} = \infty$.

Using the function $D(p_{ij}) = (w \times W_\eta)$ which returns the delivery time (at window W_η) for the given path relative to t_{min}, the *shortest temporal path length* is defined as:

$$l_{ij} = min(D(q_{ij})), \forall q_{ij} \in Q_{ij} \tag{2}$$

B. Temporal Centrality Calculation

In social network analysis, various centrality measures have been defined for different applications, however for identifying the best node to spread a message quickly, *closeness centrality* has been shown to be effective [19].

Two nodes of a static graph are said to be *close* to each other if their geodesic distance is small. In a static graph an estimation of the global *closeness* of a node i is obtained as the average static shortest path length to all other nodes in the graph, or more formally for a node i:

$$C_{\text{static}}(i) = \frac{1}{N-1} \sum_{j \neq i \in V} p_{i,j}(i) \tag{3}$$

where $p_{i,j}$ is the number of hops in the shortest path from node i to node j and N is the set of nodes in the network [21]. However, this does not capture the time-varying nature of the contacts such as time order. We can extend the definition of closeness to temporal graphs using the temporal shortest path length between nodes, which is a measure of how fast a source node can deliver a message to all the other nodes of the network. Given the shortest temporal distance $l_{ij}(t_{min}, t_{max})$, the *temporal closeness centrality* is defined as follows:

$$C_{\text{temporal}}(i) = \frac{1}{N-1} \sum_{j \neq i \in V} \frac{1}{l_{i,j}} \tag{4}$$

	CAMBRIDGE	INFOCOM	MIT
Environment	Office	Conference	Campus
N	18	78	100
Start Date	3 Feb '10	23 Apr '06	26 Jul '04
Duration	10 Days	5 days	280 days
Avg. contacts per day	1927	25796	231
Scanning Rate	30 sec	2 min	5 min

Table I
EXPERIMENTAL DATASETS

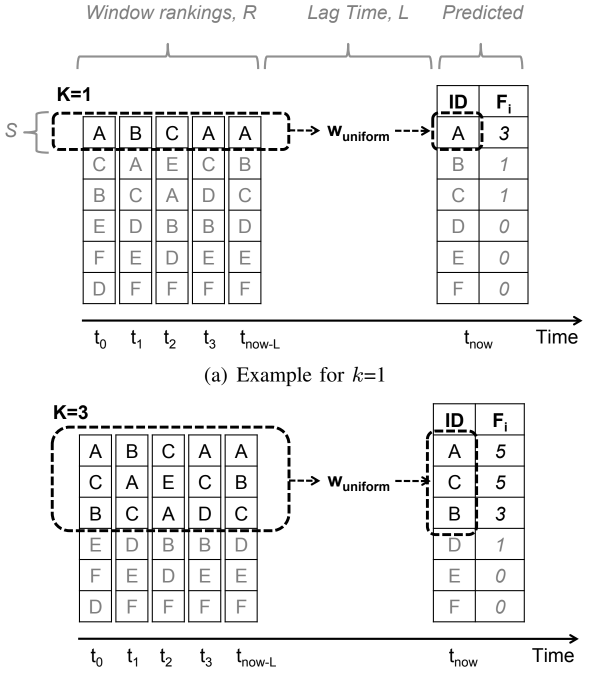
where $1/l_{i,j}$ is the inverse of the temporal path length to a destination node j, so that nodes that have *on average* shorter temporal distances to the other nodes are considered more *central*.

We compare temporal closeness with the static version using aggregated static graphs (i.e., considering a graph where an edge is present if it appears at least once in any time window in $\mathcal{G}_t$). The calculation of closeness centrality for all nodes a static graph has $O(N^3)$ time complexity, while it requires $O(N^3 + I)$ in the aggregated static graph, where I is the number of total interactions between nodes. The calculation of temporal closeness has $O(N^3\tau)$ time complexity, however, in practice, this is much smaller since most windows are sparse and the algorithm can stop once all nodes are reached. Note also that this computation can be parallelized with respect to each source node i.

C. Empirical Datasets

To evaluate our socio-temporal aware mobile malware containment scheme, three traces of real mobile device contacts carried by humans are used: Bluetooth traces of researchers at the University of Cambridge, Computer Laboratory, as part of an emotion sensing experiment [16]; Bluetooth traces of participants at the 2006 INFOCOM conference [18]; and campus Bluetooth traces of students and staff at MIT [8]. We shall refer to these as CAMBRIDGE, INFOCOM, MIT, respectively, and can be classified as office, conference and campus environments. Table I describes the characteristics of each set of traces. All three datasets were constructed from mobile device co-location where participants were given Bluetooth enabled mobile devices to carry around. When two devices come into communication range of the Bluetooth radio, the device logs the colocation with the other device. For the CAMBRIDGE dataset, all 10 days are used as part of the evaluation. For the INFOCOM dataset, since devices were not handed out to participants until late afternoon during the first day, only the last 4 days are used. For the MIT dataset, we show results for the first two weeks of the Fall semester[3] representing a typical fortnight of activity. The most important characteristic is the density, described by the average number of contacts per day. Indeed, since the INFOCOM dataset is extracted from a confined conference environment with scheduled talks, they are *temporally denser* compared to the campus and office settings.

[3] http://web.mit.edu/registrar/www/calendar0405.html



t_0	t_1	t_2	t_3	t_{now-L}
A	B	C	A	A
C	A	E	C	B
B	C	A	D	C
E	D	B	B	D
F	E	D	E	E
D	F	F	F	F

ID	F_i
A	3
B	1
C	1
D	0
E	0
F	0

(a) Example for k=1

t_0	t_1	t_2	t_3	t_{now-L}
A	B	C	A	A
C	A	E	C	B
B	C	A	D	C
E	D	B	B	D
F	E	D	E	E
D	F	F	F	F

ID	F_i
A	5
C	5
B	3
D	1
E	0
F	0

(b) Example for k=3

Figure 2. Example of the top-k set membership prediction problem, using *uniform weighted* frequency selection.

III. Predicting Central Nodes for Patch Dissemination

We now turn our attention to predicting the top-k highly ranked temporal centrality nodes based on past observations. This section is split into three parts: firstly, we describe how past rankings can be used to predict future rankings (Section III-A); secondly, we gain insight into the predictability of empirical datasets by demonstrating the correlation between rankings at different time points (Section III-B); and finally, we use these insights to inform our prediction function design (Section III-C).

A. Top-k Prediction Model

The top-k prediction model captures the problem of identifying the top-k nodes to start spreading a patch, starting from the current instant of time. Intuitively, using past observations, this prediction is based on the number of times a node i is in the set of top-k central nodes in the previous intervals of time. This frequency of observations can also be weighted by considering the time difference relative to the current time (i.e., more recent observations could have higher weighting or vice versa, etc.).

1) Example: To illustrate this idea, Figure 2(a) depicts the problem of predicting the top $k = 1$ node at the current time t_{now}. For each time window (x-axis), we construct an ordered list of node identifiers. For example, at time t_0, nodes are ranked by a centrality measure as (A, C, B, E, F, D). Since we may not have the most recent information of contacts and node rankings, we need to take into consideration a lag time L between t_{now} and the last training window at t_{now-L}.

Using this model we need to define a suitable weighting function on the top-k set of nodes in these past windows; this shall be discussed in further detail in Section III-B2, however, for now let us consider a simple *uniform* weighting function W_{uniform}, where all training windows are treated equally. Since node A appears three times across the training windows, it has the top weight and would be sent the patch.

Extending this to $k = 3$, again we iterate over all training windows, weighting the top 3 nodes accordingly. Notice again that node A is predicted to be in the top 3 nodes, along with node C and B; a patch is sent to all three devices.

Finally, it is worth noting that patching additional nodes might provide a limited benefit in some cases. For example, if two temporally connected components exist, the top-2 nodes may belong to the same component. If the infection is started also from this additional node, the benefit will be incremental, since both nodes are members of the same connected component. However, this proposed scheme does allow for *redundancy* which might be very useful given the inherent uncertainty of predictions. We shall see in Section IV-E that temporal centrality requires a smaller value of k for an effective containment scheme.

2) Definitions: More formally, given the number of top nodes k, lag time L and current time t_{now}, we first construct the temporal graph $\mathcal{G}(t_0, t_{now} - L)$ from the uploaded contact data. Next for every graph $G_t \in \mathcal{G}$ at time t, we calculate the temporal centrality C_t using $\mathcal{G}(t, t_{now} - L)$.[4] From this we construct the list of *window centrality rankings* $R(t_0, t_{now-L})$ for each time window in the interval $[t_0, t_{now-L}]$. Each window centrality ranking $r^t \in R$ at time t is an ordered list of N node identifiers ranked by temporal centrality using C_t. Next we construct the list of *top-k centrality window sets* $S^k = (s^0 ... s^{W-1})$, where set s^t corresponds of top-k centrality nodes in the window ranking r^t.

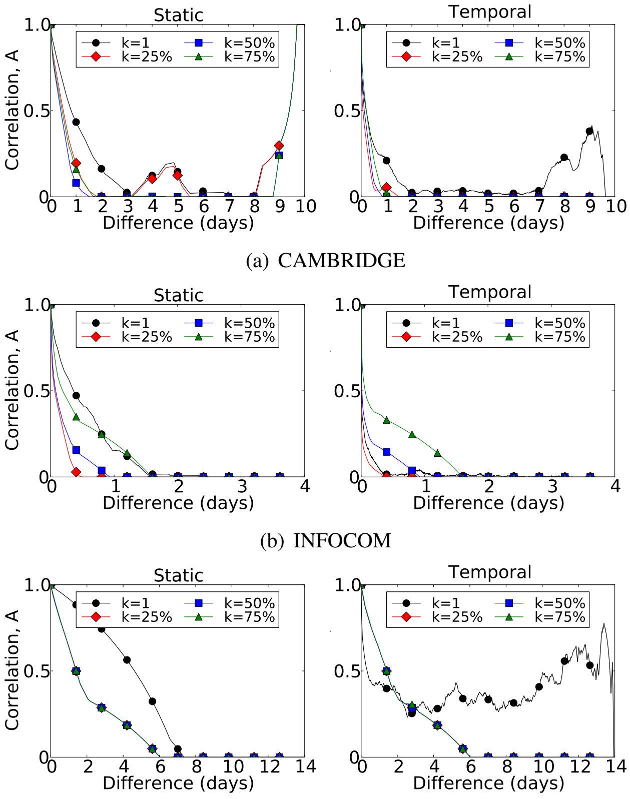
From this, given the top-k sets $S^k(t_0, t_{now-L})$, for each node i, its *weighted frequency value* F_i is defined as:

$$F_i = \sum_{t=0}^{W-1} z_i^t w(d), \qquad z_i^t = \begin{cases} 1 & \text{if } i \in s^t \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where z_i^t is used to count the presence of node i in the top-k set s^k, $d = t_{now} - t$ is the difference between the time to be predicted and the training window and $w(d)$ is an aging function used to assign different values to the presence of the node in the set of the top-k nodes in a certain window.

Then the nodes are sorted in descending order by their value of F_i and the top-k are selected for patching. In the previous example a uniform weighting $w(d) = 1$ was described. Note that although contact uploads could be

[4] It is pertinent to note that this model can be generalized to any form of node ranking.



(a) CAMBRIDGE

(b) INFOCOM

(c) MIT

Figure 3. Plotting self similarity between the rankings related to all windows a with all windows $b \leq a$, averaged by time difference $d = a - b$ (x-axis). Static (left column) and temporal (right) centralities plotted. Legacy correlation is observed with both static and temporal centrality.

staggered between different devices, we consider a uniform lag time (for example, all nodes uploaded at the same time yesterday). This is reasonable since any extra recent information increases prediction accuracy. In Section III-C we shall describe more refined prediction functions.

B. Predictability of Human Contact Traces

The derivation of our prediction functions is founded on the hypothesis that since human mobility is highly regular [6], a central person today is highly likely to be central at some point in the future.

1) Top-k Correlation Function: To test this hypothesis, we first define a *correlation function* to measure the similarity of top ranking nodes between different windows. Building on definitions in Section III-A2, given a sequence of top-k centrality window sets $S^k(t_{min}, t_{max})$, we simply use the Jaccard index between any two given window sets $s^a, s^b \in S^k$ and where $k = |s^a \cup s^b|$:

$$A^k_{a,b} = \frac{|s^a \cap s^b|}{|s^a \cup s^b|} \tag{6}$$

2) Testing for Top-k Correlations: We now measure the self-similarity between the rankings for different time windows, by first calculating the complete sequence of window centrality rankings $S(t_{min}, t_{max})$ for each dataset, and then plotting the correlation function $A_{a,b}$ for every training window $s^a \in S$ against a past window $s^b \in T$ where $b \leq a$. We repeat this for different values of k. Figure 3 plots the time difference $d = a - b$ across the x-axis against $A_{a,b}$ on the y-axis, averaged by d. First, we notice that, as expected, as we increase k the correlation function A also increases. However, we also notice across both static and temporal closeness centralities, there is a clear *legacy* effect in that top-k nodes are stable for some consecutive time windows (around a day in all traces). The peak at around 10 days in the CAMBRIDGE dataset can be attributed to the devices being collected and physically colocated at the end of the experiment. We also tested these correlations against a null model where we randomly shuffle the windows and calculate the same correlation function A: we found $< 2\%$ correlation for top-75% nodes uniformly across different time differences; for clarity, these are not plotted.

Function	$w(d)$
Uniform	1
W-log(d)	$\log(d+2)^{-1}$
W-sqrt(d)	$(\sqrt{d+1})^{-1}$
W-exp(d)	$(2^d)^{-1}$

Table II
PREDICTION FUNCTIONS WITH TIME DIFFERENCE, d.

C. Prediction Function Design

Our aim is to predict the top-k ranked nodes from which to spread the patch by taking advantage of the knowledge about the previous evolution over time of the network. By making use of past observations, this prediction is based on the number of times a node i is in the set of top-k nodes which can also be weighted by the time difference relative to the current time. Since we have observed both a strong correlation with recent past windows (in all centralities) we design empirical functions which weight past windows by distance in time.

We now describe four possible prediction functions based on a weighted average characterized by different complexity[5]. These functions are summarized in Table II. From our observations of a strong correlation with recent time windows, we can assign an age weighted function to a nodes membership in a previous time window t_i with time difference $d = (t_{now} - t_i)$: **W-log**(d), **W-sqrt**(d), and **W-exp**(d). In addition, we also compare to a simple option that weights all previous set membership equally: **Uniform**. Note that these functions can be computed in $O(M)$ for one prediction of $w(d)$ where M is the number of training time windows used.

Our approach has two key advantages: (1) it is simple to implement and deploy since we only require the past

[5] We have tested other potential alternatives, but given the space limitations, we describe those that led to the best prediction performance.

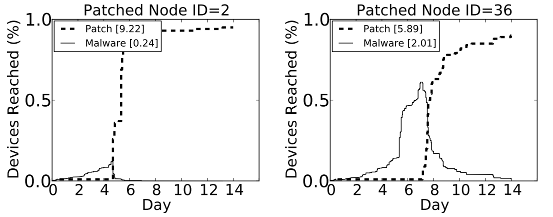


Figure 4. MIT: Malware started at day 0 and patch started 1 day later from device selected by temporal closeness (left) and random (right). Area under each curve reported in legend.

centrality values of *mobile services*, rather than tracing the whole past geometric positions of nodes; (2) it requires linear time to approximate network centrality. Our strategies are thus useful for large-scale and online computation – training data can be frequently updated in real time.

IV. Evaluation

We now evaluate STOP through *trace-driven* simulations using as input the three datasets described in Section II-C, replaying the trace contact-by-contact to simulate the spreading of a malicious worm. The top-k devices are chosen according to the calculated temporal closeness and static closeness centrality rankings[6] as described in Section III-A2 where w is set to the finest window granularity, corresponding to the scanning rate of the devices in each dataset (for example, 30 second windows for CAMBRIDGE).

The nodes that are initially infected with malicious messages are chosen uniformly randomly, averaged over 100 runs. Our evaluation is based on the following assumptions: firstly, when a node receives a patch message, it is immunized for the rest of the simulation (i.e., we assume that the malware does not mutate over time); secondly, there is always a successful file transfer between devices (errors in transmission can be taken into consideration in the assessment of the contention scheme without changing significantly the results of our work, assuming random transmission failures); thirdly, following the previous point, we assume 100% malware infection rate (this gives us the worst case scenario); finally, an attacker chooses nodes at random (or users download malware randomly).

A. Parameters and Evaluation Metrics

Figure 4 plots the infection rate when a piece of malware is started at midnight on the first day of the MIT dataset and a patch is sent 1 day later from two different devices. Note the difference between the device selected by temporal closeness (left) and a random selection (right) in containing the malware, which indicates that there is a clear advantage in using such a temporal graph measure. To understand if this still holds across different malware start times and patch delays we first characterize the infection rate using three metrics:

- the area under the curve (AUC), which captures the behavior of the infection over time with respect to the number of infected devices;
- the *time zero* in hours necessary to achieve total malware containment (TZ); and
- the peak number of compromised devices ($MAXN$).

Note that we aim to *minimize* these three metrics. In the figure, device 2 reduces the AUC of the malware to 0.24 vs. 2.01; can contain the malware quicker (6 days vs. 2 weeks) and has a smaller $MAXN$ (10% vs. 60%). Since there are many parameters under investigation, these three metrics now help to understand the effects of:

- Malware Start: the time at which malware is deployed, starting every 3 hours of each trace;
- Patch Delay: the delay before a patch is ready to be deployed from $\{1$ hr, 3 hrs, 24 hrs, 48 hrs$\}$;
- Upload Interval: the frequency of mobile device contact uploads $\{1$ hr, 24 hrs$\}$;
- Initial number of compromised devices and number of devices we start a patch from.

B. Effect of Malware Start Time

Figure 5 plots the effects of disseminating malware starting from a single device at different times (x-axis) during the trace for the MIT dataset. We fix the upload interval to 1 day and average across all delay times. For each centrality measure (as described in Section II-B), each plot shows how different prediction functions perform when selecting a single nodes to start spreading the patch. We plot curves for naive random patch device selection and an oracle device selection, corresponding to the case of temporal closeness with knowledge of all future contacts which has previously been shown to be the most effective for opportunistic patch dissemination [19]. To compare between curves, we present the area under each curve in the legend.

First, notice that there is a significant improvement over a random node selection and that the performance of devices selected approaches that of the oracle. Second, notice that both static and temporal centrality are highly accurate across all prediction types, however, static centrality is only accurate when using W-exp. Third, comparing the best prediction function between static and temporal centralities, temporal has a lower AUC, translating to better patching performance across different start times. Fourth, all methods take around 150 hours (6.25 days) to fully contain the malware, however, notice that there is less than 10% MAXN when using W-log weighting for static and temporal centralities: this fits our aim of spreading the patch to as many nodes quickly and rely on the natural chain of human contacts to eventually trickle the patch to remaining devices over time. Finally, common across all centrality types, there are small peaks around noon for TZ and $MAXN$ and troughs during the evening, which demonstrates that a time-aware approach is required since malware has more opportunity to spread during the daytime.

[6] In the case of the static model, C_t is the static centrality calculated on a graph aggregated over $\mathcal{G}(t, t_{now} - L)$.

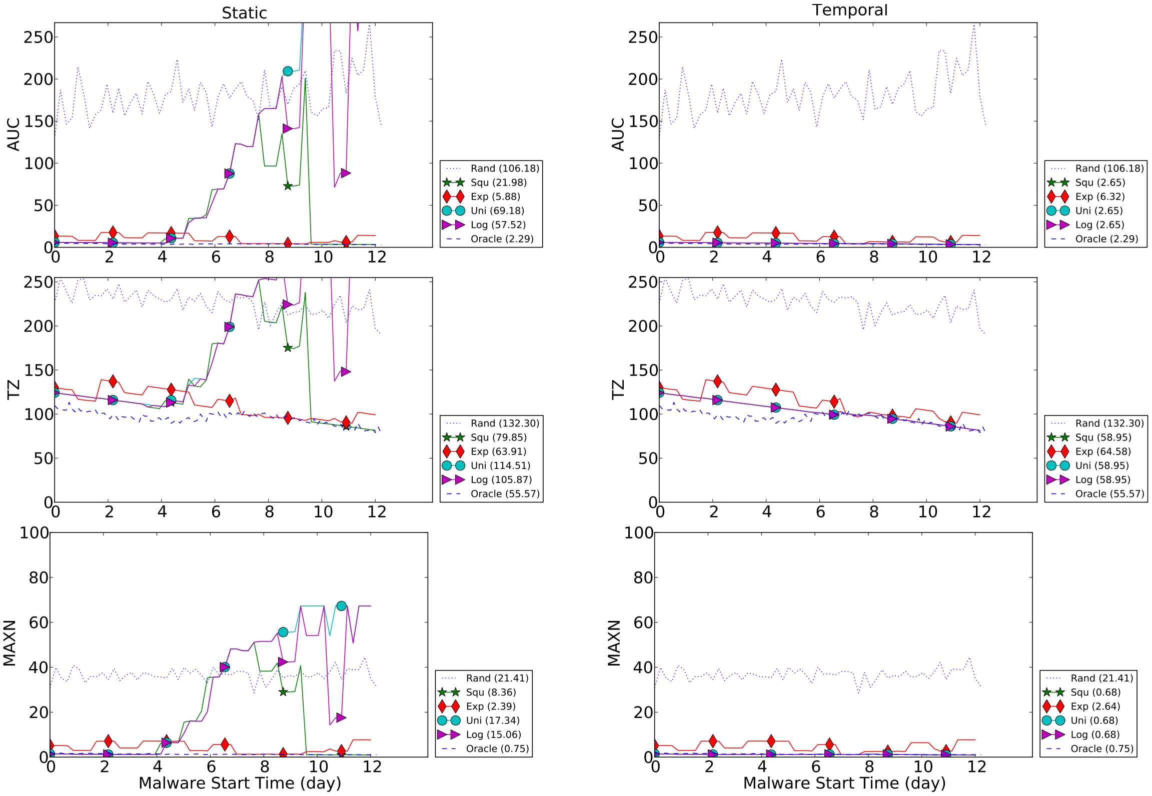


Figure 5. MIT Traces: Comparison of Centrality type vs. Prediction Function, as a function of Start Time (x-axis).

Model	CAMBRIDGE		INFOCOM		MIT	
	Static	Temporal	Static	Temporal	Static	Temporal
Uniform	9.06	6.27	8.85	17.65	69.18	2.65
W-Exp	**6.12**	6.38	12.76	**10.31**	**5.88**	6.32
W-log	9.06	**6.07**	**8.84**	17.71	57.52	**2.65**
W-Squ	9.13	6.07	9.42	17.7	21.98	2.65
Best	W-Exp	W-Log	W-Log	W-Exp	W-Exp	W-Log
Oracle	3.37		1.52		2.29	
Overhead	1.82x	1.80x	5.82x	6.78x	2.57x	1.16x

Table III
COMPARING CENTRALITY VS. PREDICTION FUNCTION, MEASURED BY AUC OF ALL START TIMES AVERAGED OVER ALL LAG TIMES.

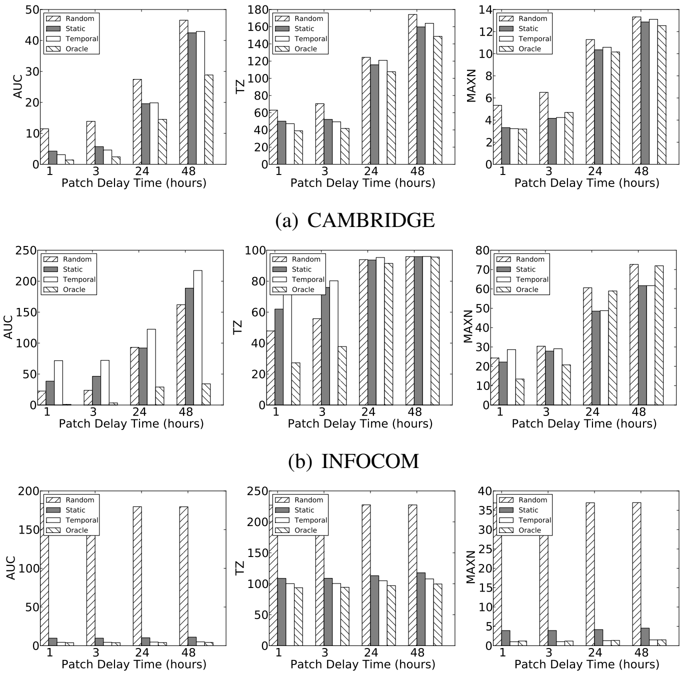
Generalizing to all traces, we enumerate in Table III the AUC for all centrality prediction pairs for all datasets. There is no single perfect choice prediction function that is best for all centralities, however, the centrality prediction pairs which minimize the AUC can be used as a first approximation (shown in bold). Note that there is more than one best prediction function for temporal centrality in the MIT dataset, however, we use W-log since it is the best performing overall for temporal centrality across all datasets. Now, comparing the best performing centrality prediction combination between datasets, we observe that the temporal approach performs best to minimize AUC in CAMBRIDGE and MIT datasets, however, static centrality has more accurate prediction for INFOCOM. This suggests that in confined spaces with denser contacts a static model may be best suited; however, temporal can still be relied on across all scenarios to contain malware in a finite time and in most cases perform better than static centrality. Quantifying the overhead of the best centrality prediction combination with the oracle, using temporal centrality we can achieve up to 1.16x accuracy in the best case and also on average across the three scenarios temporal centrality has the lowest overheads.

C. Increasing Patch Delay

Figure 6 plots the best centrality prediction pairs, binned by increasing patch delays (x-axis), for the CAMBRIDGE, INFOCOM and MIT datasets, respectively. Increasing the patch delay is detrimental to malware containment, increasing the AUC, time of total containment and peak infected devices. Across all datasets, this is most prominent in the conference (INFOCOM) environment which again can be attributed to the confined space which increases the malware spreading rate and again suits a static model better than temporal centrality. However, for CAMBRIDGE and MIT, temporal centrality outperforms static device selection.

D. Effects of Contact Upload Interval

So far we have considered a daily upload interval; Figure 7 again plots an increasing patch delay for the CAMBRIDGE dataset (compared with Figure 6(a)) but for an hourly upload interval. We can note two things: firstly, there is very little improvement from a daily upload, and secondly, static methods have improved more than temporal. This suggests that these prediction functions are still able to



(a) CAMBRIDGE

(b) INFOCOM

(c) MIT

Figure 6. Best centrality prediction binned by patch delay (upload interval 24 hours).

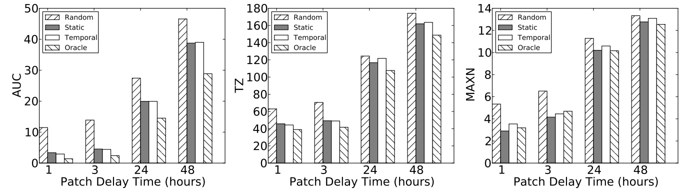


Figure 7. CAMBRIDGE: Best centrality prediction binned by patch delay (upload interval 1 hour).

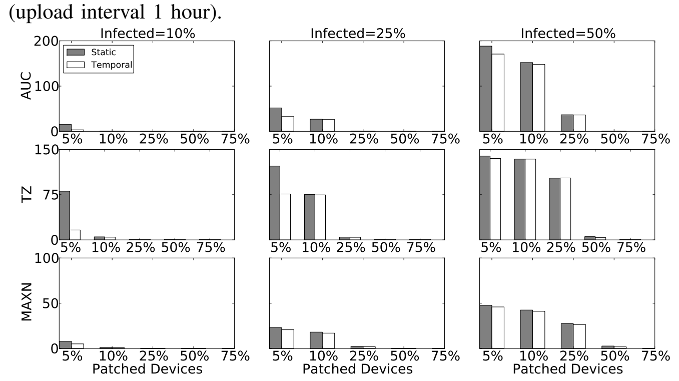


Figure 8. MIT: Effects of increasing number of patched devices against initial infected devices. Midday infection, 3 hours patch delay.

perform accurately even with missing data (increased lag time, L).

E. Varying initial compromised and patched devices

To understand the effects of increasing the number of initially infected devices I_n and increasing top-k patched devices, we fix the malware start time to day 2 at midday (the most damaging time of day for malware spreading), upload time to 1 hour and patch delay to 3 hours. Figure 8 plots the percentages of initially infected devices (equal to 10% (left column), 25% (middle) and 50% (right)) with increasing top-k patched devices (x-axis) using the MIT dataset.

Starting with a low $I_n = 10\%$, containment is effective with an equally low value for $k = 5\%$. Increasing the number of devices to spread the patch past $k = 10\%$ does not add performance gains. Notice that temporal centrality is able to contain the malware within $TZ = 10$ hours, compared with the static one, which take around 75 hours.

With $I_n = 25\%$, again temporal offers advantages using a low value of k. However, as we increase the value of k to 10% then static and temporal centrality are very similar; this is more apparent when $I_n = 50\%$. From this, we observe that temporal can more effectively select a smaller set of devices compared to static methods. This is a useful property, because one of the requirements is the minimization of the number of devices required to receive the initial patch.

V. Discussion and Comparison with the State of the Art

We have shown the benefits and limitations of STOP: by using real traces, we have demonstrated that STOP can contain malware in a finite time in three different types of environments (office, campus and conference), however, in presence of very temporally dense networks (e.g., a conference environment) a model based on static metrics measured on aggregate networks is worth considering. Past work has shown that with global knowledge of all contacts temporal metrics are more accurate than static ones in different environments [19], but with dense traces the lag time L (i.e., any missing training windows) is key to accurately identifying the same devices with only historical knowledge. Also, compared to static measures of node importance, we found that temporal centrality is more accurate at selecting a smaller subset of devices to start opportunistic dissemination of such a patch. Indeed, there is a balance of accuracy and complexity; static centrality is easier to calculate, however, temporal centrality is better at predicting high-impact nodes to disseminate a patch. At the same time, since computation is offloaded to a mobile network operator, such a requirement can be easily fulfilled. We evaluated several intuitive prediction functions and found that even simple prediction functions can be very effective for malware containment.

We are aware of the privacy implications of the proposed approach. However, we would like to point out that since mobile network operators already collect the location of devices (i.e., through cell tower registrations for billing purposes) additional information on short range radio contacts does not significantly intrude on user privacy; future work could take advantage of this information to eliminate the need for devices to collect contact information. In addition, actual contacts can be deleted after the centrality rankings are calculated since raw contacts are not required for later prediction. Also, potentially an attacker could use the same temporal centrality prediction system to distribute its mobile worm, however, an attacker will not be able to collect the same amount of contact data and hence their prediction will be limited by incomplete data.

There are still areas open for investigation. Firstly, we assume total infection and patch success, whereas in real life this depends on many factors such as contact time; our study is clearly a worst case scenario. Secondly, from the simulation results, the best prediction function can be improved depending on the nature of network environment; our aim is to further explore automatic derivation of an optimal weight assignment for a given dataset.

Social network based strategies for patch propagation have received increasing attention given the availability of data about physical and virtual user interactions. For example, Zhu et al. [22] propose that the most central nodes derived from phones call logs should be prioritized for patching. However, this only captures potentially long-distance relationships and misses important opportunistic contacts that Bluetooth worms can exploit. Zyba et al. [23] evaluate the spreading of a patch via short-range radio transmission. However, this work is based on a random mobility model and assume homogeneous mixing and degree distribution over time. As we have shown, mobile phone contact networks are driven by periodic human schedules and so the models proposed in this paper could be considered as an over-simplification of real world situations. Such schemes are also partially founded on work into robustness of networks against random failures and targeted attacks of individual nodes in complex networks [5]: these solutions, however, are based on static graph representations which ignore time ordering and frequency of contacts. More recently, we considered temporal relationships based on dynamic networks in order to effectively select a set of nodes as a starting points of the patching process [20], [19]. We demonstrated the feasibility of time-aware central node identification methods for patch dissemination but this strategy required a priori knowledge of future contacts, which is not available in practice. Instead, in this work we propose a *predictive* socio-temporal aware central node identification method that only requires past history of device contacts.

VI. Conclusions

We have presented STOP, a socio-temporal aware dissemination scheme based on the identification of the key nodes in a mobile network from which a patching process should start to contain malware effectively. Through extensive simulations we have demonstrated that STOP can block malware spreading in a finite time. We have found that temporal centrality metrics with a logarithmic-weighted function is well suited to predicting the best nodes to start spreading an opportunistic patch compared to schemes based on static centrality measures.

Acknowledgements

This work was funded in part through EPSRC Project MOLTEN (EP/I017321/1).

References

[1] Virus description: Bluetooth-worm:symos/cabir. http://www.f-secure.com/v-descs/cabir.shtml, 2004.

[2] Virus description: Backdoor:wince/phonecreeper.a. http://www.f-secure.com/v-descs/backdoor_wince_phonecreeper_a.shtml, 2009.

[3] Virus description: Sk.flexispy.h. http://virus.netqin.com/en/symbian/SK.Flexispy.H/, 2009.

[4] Virus description: Trojan-spy:symbos.zbo.b. http://www.f-secure.com/v-descs/cabir.shtml, 2011.

[5] R. Albert, H. Jeong, and A. Barabási. Error and attack tolerance of complex networks. *Nature*, 406(6794), July 2000.

[6] A. Clauset and N. Eagle. Persistence and periodicity in a dynamic proximity network. In *Proc. of DIMACS '07*, 2007.

[7] F. Cohen. Computer Viruses: Theory and Experiments. *Computers and Security*, 6(1):22–35, 1987.

[8] N. Eagle and A. Pentland. Reality Mining: Sensing Complex Social Systems. *Pers. Ubiq. Comput.*, 10(4):255–268, 2006.

[9] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In *Proc. of OSDI*, 2010.

[10] W. Enck, P. Traynor, P. McDaniel, and T. L. Porta. Exploiting open functionality in SMS-capable cellular networks. In *Proc. of CCS*, 2005.

[11] T. Hossmann, T. Spyropoulos, and F. Legendre. Know thy neighbor: Towards optimal mapping of contacts to social graphs for DTN routing. In *Proc. of IEEE INFOCOM*, 2010.

[12] M. Hypponen. F-secure weblog: The grand opening! www.f-secure.com/weblog/archives/00000568.html, 2005.

[13] M. Hypponen. Malware goes mobile. *SCI. AM.*, Nov. 2006.

[14] M. Landman. Managing smart phone security risks. In *Proc. of ACM InfoSecCD '10*, 2010.

[15] M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel. Semantically rich application-centric security in android. In *Proc. ACSAC '09*, 2009.

[16] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucinas. EmotionSense: a mobile phones based adaptive platform for experimental social psychology research. In *Proc. of UbiComp '10*, Sept. 2010.

[17] M. R. Rieback, P. N. Simpson, B. Crispo, and A. S. Tanenbaum. RFID malware: Design principles and examples. *Pervasive and Mobile Computing*, 2(4):405–426, Nov. 2006.

[18] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau. CRAWDAD data set cambridge/haggle (v. 2009-05-29), May 2009.

[19] J. Tang, C. Mascolo, M. Musolesi, and V. Latora. Exploiting temporal complex network metrics in mobile malware containment. In *Proc. of IEEE WoWMoM '11*, 2011.

[20] J. Tang, M. Musolesi, C. Mascolo, and V. Latora. Temporal distance metrics for social network analysis. In *Proc. of ACM WOSN '09*, 2009.

[21] S. Wasserman and K. Faust. *Social Networks Analysis*. Cambridge University Press, 1994.

[22] Z. Zhu, G. Cao, S. Zhu, S. Ranjan, and A. Nucci. A social network based patching scheme for worm containment in cellular networks. In *Proc. of INFOCOM*, 2009.

[23] G. Zyba, G. M. Voelker, M. Liljenstam, A. Mehes, and P. Johansson. Defending mobile phones from proximity malware. In *Proc. of INFOCOM*, 2009.