Closing the Pandora's Box: Defenses for Thwarting Epidemic Outbreaks in Mobile Adhoc Networks

Rahul Potharaju, Endadul Hoque, Cristina Nita-Rotaru
Department of Computer Science,
Purdue University,
West Lafayette, IN, USA
Email: {rpothara,mhoque,crisn}@cs.purdue.edu

Saswati Sarkar, Santosh S. Venkatesh
Department of Electrical Engineering,
University of Pennsylvania,
Philadelphia, PA, USA
Email: {swati, venkates}@seas.upenn.edu

***Abstract*—The openness of the Android operating system increased the number of applications developed, but it also introduced a new propagation vector for mobile malware. We model the propagation of mobile malware using epidemiology theory and study the problem as a function of the underlying mobility models. We define the optimal approach to heal an infected system with the help of a set of static healers that distribute patches, as the T-COVER problem and show that it is NP-HARD. We then propose two families of healer protocols that trade-off time recovery and energy consumed by sending patches. The first one uses randomization to ensure a small recovery time but may result in healers sending more patches than needed. The second one uses system feedback to optimize energy consumed by sending patches, but it may result in a larger recovery time. We show through simulations using the NS-3 simulator that despite lacking knowledge of the future, our protocols obtain a recovery time within a 10x bound of the oracle solution that knows the arrival time of the infected nodes.**

I. INTRODUCTION

With the advent of Google's Android, the number of wireless devices with complex capabilities and supporting open source operating systems has significantly increased. While the openness of operating systems induces developers' motivation, it also introduces a new propagation vector for mobile malware. Recent reports show a significant increase in malware targeting Android devices [1, 2].

Significant research focused on propagation modeling, detection, and application profiling of malware in the context of wired networks [3]–[7]. Those results do not model mobile malware which spreads directly from device to device by using short-range communication such as WiFi or Bluetooth. Mobile malware propagation has been studied using mean field compartmental models [8] which assume that each infected node will contact every neighbor once within one time step, *i.e.*, the infectivity is equal to the connectivity. Such models do not take into account that mobile malware does not spread at an even contact rate, as spreading requires devices to be within each other's proximity which in turn depends on user mobility. Most previous research on mobile malware has either not considered mobility [9]–[11] or has given limited considerations to it [12, 13]. Approaches that considered mobility have used popular models like the random waypoint model which, as it has been shown, does not realistically mimic human mobility [14].

While there has been work studying mobile malware propagation, the problem of *infection containment* in wireless networks was less studied. The work of [15] analytically studied containment of infection in a mobile network through countermeasures such as reducing communication range of nodes during an infection outbreak. The work does not consider realistic mobility models and does not propose concrete protocols to deploy and activate such countermeasures. The work in [16] introduced replicative and non-replicative patch disseminations assuming a network cost function and proved that the dynamic control strategies have a simple optimal structure. However, the impractical determination of the healer activation time and the lack of inclusion of the resource cost incurred by each patch dissemination make the techniques difficult to apply directly to energy constrained realistic scenarios.

In this paper, we take the first step towards designing countermeasures for malware propagation under the presence of realistic mobility in a practical scenario. We investigate the dependence of infection spread on the underlying mobility model in order to systematize the design of countermeasures. We introduce the concept of *healers* to mimic the recovery process in a standard epidemic model and we focus on *static healers*, i.e., *immobile* healers, which represent a realistic model because they can be directly mapped to real-world scenarios. For instance, static healers can be considered as cellular base stations (where no two stations cover the same cell in most cases) and the mobile nodes can be considered as users carrying mobile phones (moving with a certain mobility model). Unlike [12], our static-healers are not white-worms and do not deactivate infected nodes. Our contributions are:

- We show that the infection spread in mobility models that mimic human behavior is slower than standard mobility models due to different contact rate and spatial distribution characteristics. We compare the Truncated Levy Walk (TLW) and Random Waypoint (RWP) mobility models and show that the epidemic spread in TLW is *relatively slower* compared to RWP. This finding indicates that when designing countermeasure mechanisms, the time constraints are less tight than believed and that time-dependent assumptions can be relaxed to some extent, resulting in relatively lower consumption of energy.
- We model countermeasures to malware spread using static

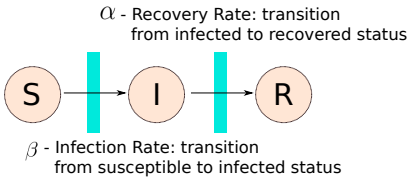


Fig. 1. SIR Model: S, susceptible; I, infected; R, recovered

healer nodes. Static healers once placed in the area, act independently to deploy a patch when they sense nodes in their proximity. A healer-based solution optimizes: (i) the time it takes to heal the entire system by patching all the infected nodes and (ii) the total number of patches broadcasted. We formulate the optimal solution based on static healers as a T-COVER problem and show that the T-COVER is NP-HARD.

- We design ORACLE, a $log(n)$ greedy approximation algorithm that computes the optimal healing time knowing the placement of the static healers and *the future*, i.e. the exact time instances when the infected nodes arrive within each of the healer's proximity.
- We propose a novel healer placement strategy using blue-noise distribution generating Poisson Disk Sampling. We show that unlike random placement that results in many overlapping healers which cover the same area, our method allows healers to cover disjoint areas, thus enabling them to independently cover more infected nodes.
- We design two families of healer protocols, RH and PH, that trade-off time recovery and energy consumed by sending patches. RH uses randomization to ensure a small time recovery but may result in healers sending more patches than needed. PH uses system feedback to optimize the energy consumed by sending patches, but it may result in a larger time recovery. We compare our protocols with the ORACLE protocol and show through simulations that despite lacking knowledge of the future, both RH and PH protocols obtain a recovery time within a 10x bound of the oracle.

The rest of this paper is organized as follows. §II describes our system model, our assumptions and introduces the mobility models used in this paper. §III analyzes the infection dynamics as a function of the underlying mobility models. §IV introduces our healers and §V provides simulation results. §VI discusses related research and §VII concludes the paper.

II. SYSTEM MODEL

In this section, we construct a framework for analyzing the propagation of malware over a mobile ad hoc network that relies on epidemic theory to capture both the spatial interaction of nodes and the temporal dynamics of infection propagation.

A. Mobility Models

Due to the difficulties in adapting real-trace data to long running simulations [11], we decided to use analytical models derived from real-trace data instead. Specifically, we use the *Random Waypoint* (RWP) and *Truncated Levy walk* (TLW) mobility model to generate synthetic mobility traces. We selected RWP because it is a typical mobility model used to study mobile malware propagation. We selected TLW because it was shown to mimic human mobility. Unless otherwise noted, we use a node velocity of 0.6 m/s to mimic low velocity realistic human mobility in both mobility models throughout the paper.

Random Waypoint (RWP): RWP is a widely used mobility model [17]–[19] and includes pause times between changes in direction and/or speed [20]. A mobile node begins by staying in one location for a certain time period (pause time). Once this time elapses, the mobile node chooses a random destination in the simulation area and a speed that is uniformly distributed between $[v_{min},\ v_{max}]$. The mobile node then travels toward the newly chosen destination at the selected speed. Upon arrival, the node pauses for a specified time period and starts the whole process again (see Fig. 2(a)).

The initial random distribution of mobile nodes is not representative of the manner in which nodes distribute themselves when moving as the instantaneous mobile node neighbor percentage possess high variability [21]. We use the approach suggested by [22] and discard the initial 1000 seconds of simulation time produced by RWP in each simulation trial.

Truncated Levy Walk (TLW): Rhee *et. al.* recently [14] reported that human walks performed in outdoor settings of tens of kilometers resemble a truncated form of Levy walks commonly observed in animals such as spider monkeys, birds and jackals. A *Levy walk* is a type of random walk in which the increments are distributed according to a heavy-tailed probability distribution, *i.e.*, their tails are not exponentially bounded. The distribution used is a power law of the form $y = x^{-\alpha}$ where $1 < \alpha < 3$. TLW is a random equivalent mobility model for human walks in that it can describe some important characteristics of human walks (e.g. flight length, pause time and inter-contact time) despite being a random model. Intuitively, Levy walks consist of many short flights and exceptionally long flights that nullify the effect of such short flights (see Fig. 2(b)).

B. Infection and Recovery Models

We adapt two classic epidemic models (SI and SIR) to take into account mobility. First we give a brief overview of the SI and SIR models, then describe how we use them to model malware propagation and node recovery in a mobile network.

SI Model. The SI-model is a two-state compartmental epidemic model, *i.e.*, a node can stay in one of two states: *susceptible* and *infected*. A susceptible node is vulnerable and can be exploited to be infected which in turn can infect other susceptible nodes. In this model, once a susceptible node is infected, it stays that way. The parameter that characterizes the model is the infection rate, β.

SIR Model. The SIR Kermack-McKendrick model [23] assumes that an infected node can be recovered. Specifically a node can be in one of the following states: *susceptible*, *infected*, and *recovered*. Nodes flow from the susceptible group to the infected group and then to the recovered group [24] as shown in Figure 1. The model is characterized by two parameters, the infection rate β and the recovery rate α.

Mobile Infection Model. The SI model makes the unrealistic assumption that each infected node will contact every neighbor once within one time step, *i.e.*, the infectivity is equal to the connectivity. To take into account mobility, we assume the nodes are moving according with a mobility model and we define infection spread as a function of a parameter c which we call the *probability of successful transmission*. At each time step, for every node X, we find the neighbors of X that are capable of infecting X. For each of these neighbors, we generate a random number from a uniform distribution between $[0, 1]$ and if this value is smaller than c, then X becomes infected.

Mobile Recovery Model. We adapt the SIR epidemic model as follows. Infection is modeled as in the mobile infection model above. We map node recovery through a healer that will change the state of an infected node to recovered through a healing mechanism. Once recovered, a node can no longer be infected, thus if no new nodes are added the infection will eventually disappear. The healing mechanism is distributed through a patch, a healer can send at most once during an interval of time called *epoch*, denoted as τ. We assume that healers are static, resource constrained, and act independently. We assume that there is no packet loss but note that it is straightforward to extend our model to a model having packet loss.

This model is characterized by the way the healers are placed and by the frequency with which they send patches. All healers are activated once the number of infected nodes in the system reached a system-wide parameter.

III. Infection Dynamics

In order to understand the infection dynamics of the two mobility models, we first describe our methodology and then explain the results that we observed.

A. Methodology

We use the infection model described in previous section with the parameter that controls the infection rate, $c = 0.3$ [25] to mimic a more realistic infection scenario where infection spreads slowly. We generate RWP traces by using [26] and TLW traces by using the algorithm outlined in [14]. We perform our simulations using NS-3 [27]. We simulate the behavior of a system with 100, 200, and 300 nodes in a fixed area. All results have been averaged over ten simulation runs.

We define an *inversion point* to be the time instant where 50% of the population is infected. We use this metric to indicate the *first* point in time where the number of infected nodes surpasses the susceptible ones, thus *inverting* the scenario. Intuitively, an inversion point characterizes how fast the infection is propagating in an epidemic system.

B. Results

Figure 2 shows the infection dynamics in RWP and TLW mobility models. Observe that the inversion point for RWP occurs quite early in the simulation (Fig. 2(c) indicates a time around 500 seconds) in comparison with TLW (Fig. 2(c) indicates times between 1500-3000 seconds). This indicates that the time required to infect the system is far less in case of RWP differing almost by a factor of 3 from TLW. To the best of our knowledge, this phenomenon has not been observed before as most earlier research [12, 13] has studied these mobility models in isolation. As protocols are to be designed mostly for realistic mobility models (TLW in this case), this comes as a good news in that certain assumptions such as time-constrained-ness of a protocol can be relaxed to some extent, resulting in relatively lower consumption of energy.

We gain insights into the reasons behind the slow infection propagation for TLW by using two metrics: (i) contact rate, and (ii) spatial distributions of node mobility.

1. Contact Rate: Contact rate is the average number of nodes encountered by any given node over the duration of simulation. We plot an empirical cumulative distribution curve (ECDF) of the contact rate in Fig. 2(d) for RWP and TLW. Observe that the median contact rate of nodes in case of RWP is almost always higher than that in TLW. The same effect can be observed for the 95^{th} percentile indicating that in RWP, a given node comes in contact with a relatively higher number of nodes thereby increasing its chances of infecting other nodes or getting infected by other infected nodes.

2. Spatial Distributions: The spatial distribution (*i.e.*, frequency of visits in the simulation area) of the mobility models reveals another reason behind the slow infection propagation. In order to evaluate the spatial distribution of infected nodes that move according to each of the models, we take an approach similar to [28]. Specifically, we divide the simulation area into small size cells (e.g., divide a 1000x1000m^2 into 20x20m^2 size cells) and characterize each one of them using a histogram that captures the duration of how long an infected node stays in a particular cell. We end the simulation after 50,000 seconds.

Fig. 3(a) shows the resulting spatial distribution and contour lines for a particular simulation run using RWP. We observe that the spatial distribution has a peak in the middle of the area, i.e., an infected node is most likely to be found in the central cells of the simulation area and the probability that a node is located at the border of the area goes to zero. Fig. 3(b) shows the spatial distribution and contour lines for TLW. Observe that the non-homogeneous behavior seen in the case of RWP is absent in the case of TLW, i.e., TLW exhibits a homogeneous spatial distribution. The reason for the non-homogeneous behavior in RWP is well known [28]–[30]. In short, RWP chooses a uniformly distributed destination point rather than a uniformly distributed angle. This means that nodes located at the border of the simulation area are very likely to move back toward the middle of the area. However, this is not the case as per the original definition [14] of TLW. Under a TLW, at the beginning of each step, an infected node chooses a direction randomly from a uniform distribution of angle within $[0, 2\pi]$, a finite flight time randomly based on some distribution, and its flight length and pause time from some chosen probability distributions. In the long run, the positions of the random walker (infected node in our case) has been shown to converge to another distribution, called the

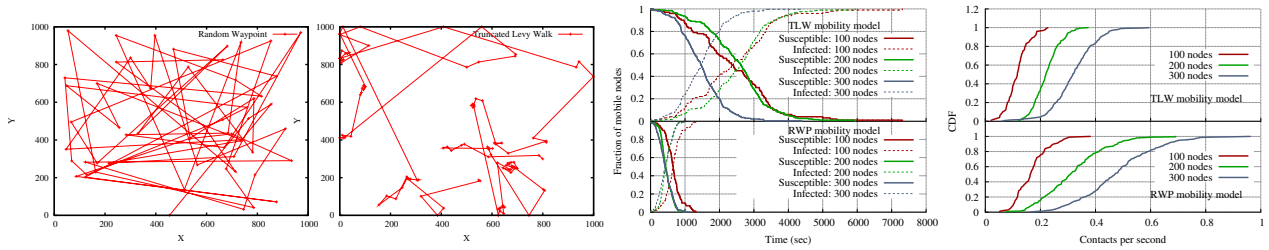


Fig. 2. (a),(b) **Tracing the path of a single node for RWP and TLW:** Observe the short paths in RWP and bursty long paths in TLW, (c) **Inversion point in RWP and TLW:** Observe that the infection spread is slower in TLW. (d) **Explaining the slow propagation:** Observe that contact rate in TLW which is less than RWP.

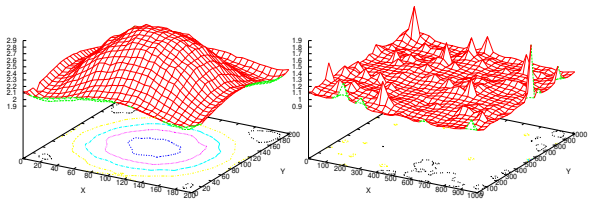


Fig. 3. **(a) Spatial Distribution of RWP:** The non-homogeneous distribution of node mobility indicates the center to be the most frequented region. **(b) Spatial Distribution of TLW:** The nearly-homogeneous distribution of node mobility indicates that all regions are equally frequented.

Levy stable distribution, which leads to super-diffusive paths, thus making the infected nodes cover the area in a nearly homogeneous manner.

In summary, in case of RWP, depending on the origin of the infection, the spread can progress rapidly because most nodes have to pass through a common point in the center which also explains why the contact rate of the nodes is higher than that in TLW. In case of TLW, due to the underlying homogeneous behavior, the rate of infection propagation is nearly the same irrespective of the point of origin of the infection.

Impact on the design of countermeasures:

- **Static healers placement**: In case of RWP, positioning a few static healers somewhere near the center of the field in a non-overlapping manner should suffice because most nodes will traverse the central point in the field anyways. However, this is not the case for TLW, because the node distribution is uniform across the field, thus requiring a way to optimize healer placement such that they cover as much field as possible.
- **Healer patch dissemination**: In case of designing a healer for TLW, having a higher patch dissemination rate will result in a lot of patches being delivered to the same set of nodes since due to the low velocity (and thus low contact rate) many nodes may continue to stay within the proximity of the healer. Therefore, for a system optimizing energy, healer patch dissemination is a function of the contact rate (details in § IV-E).

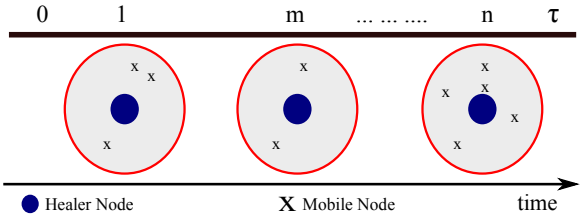


Fig. 4. **Healer Activation Problem:** Without information about its future states, predicting when to broadcast a patch to optimize the healing time is hard.

IV. Defense Protocols Based on Static Healers

In this section, we discuss defense protocols against mobile malware. We first present the problem definition, formally define the static optimal healer activation problem, show it is NP-hard, and design a greedy approximation algorithm. We discuss strategies for healer placement and present two families of static healers heuristics: randomized and profile-based.

A. Problem Definition

Healers have the ability to broadcast a patch periodically at every epoch τ. We consider the decision problem of when the healer node should be activated (i.e. switched on) within this time period to deliver a patch, to optimize along two dimensions: (i) the time it takes to heal the entire system, and (ii) the total number of patches broadcasted.

We assume that healers can sense the number of neighbors surrounding them but cannot determine which of the nodes are infected/susceptible/recovered. Note that this increases the complexity of the problem significantly. Consider the example in Fig. 4. At time slot 1, if the healer decides to utilize its patch, it will heal three nodes whereas at time slot m, it can heal only two nodes and at time slot n, it can heal five nodes. An oracle that has access to the future will pick a time slot that will make an effective use of the patch to heal the maximum number of nodes (in this case, time slot n). However, in practice, the future is not available to healer nodes.

We ask the questions: *What is an effective strategy for positioning the static healers so that two healers will avoid healing the same set of infected nodes?* and *How does the healer decide whether it should deliver the patch or wait in anticipation of a higher number of nodes in the future?* Without loss of

generality, we consider the energy consumption in delivering the patch is much higher than any other communication activity initiated by a healer. Intuitively, we are solving the problem of effectively distributing a patch without knowing the arrival distribution of infected nodes.

B. Design of an Oracle Optimal Healer

In the following, we formally define the static optimal healer activation problem, show it is NP-hard, and design instead a greedy approximation algorithm.

Let us call the task of designing a strategy for an optimal healer as the T-COVER problem.

Definition 1. *(*T-COVER*). Given a system $I_t = (\mathcal{I}, \mathcal{T})$, where $\mathcal{I}$ be the set of all infected nodes and $\mathcal{T} = \bigcup_i T_i$, where each T_i is the set of infected nodes seen by all the healers at time instance i. Let no two healers exist within the range of each other, and that a patch from a healer can heal all infected nodes within its range and will consume one time unit. The* T-COVER *problem finds a set $W \subseteq \mathcal{T}$ with min cardinality so that it covers the entire set of infected nodes $\mathcal{I}$ with $|W|$ patches.*

For example, let $\mathcal{I} = \{1, 2, 3, 4\}$ and $\mathcal{T} = \{T_1, T_2, T_3\}$ where $T_1 = \{1\}$, $T_2 = \{1, 2, 3\}$ and $T_3 = \{3, 4\}$ be the sets of infected nodes seen by the healer, then the T-COVER is $W = \{T_2, T_3\}$ meaning that a patch should be deployed at times $t = 2$ and $t = 3$ for optimality.

We now prove that the T-COVER problem is NP-HARD by reduction to the minimum set cover problem. First we state the minimum set cover problem.

Definition 2. *(*MIN SET COVER *(MSC)). Let $\mathcal{S} = \{S_1, S_2, ..., S_m\}$ be a collection of finite sets, where S_i's elements are drawn from a universal set $U = \bigcup_{i=1}^{m} S_i$. The MSC of $I_s = (U, \mathcal{S})$ is a set C with min cardinality where $C \subseteq \mathcal{S}$ and $\bigcup_{S_i \in C} S_i = U$.*

For example, assume $U = \{1, 2, 3, 4, 5\}$ and $\mathcal{S} = \{S_1, S_2, S_3, S_4\}$, where $S_1 = \{1, 2, 3\}$, $S_2 = \{2, 4\}$, $S_3 = \{3, 4\}$ and $S_4 = \{4, 5\}$. The MSC is $C = \{S_1, S_4\}$. The MSC problem is NP-HARD. Consequently, the following is entailed.

Theorem. *The* T-COVER *problem is* NP-HARD.

Proof: We prove the theorem by providing a polynomial time reduction from the NP-HARD MSC problem. Consider $I_s = (U, \mathcal{S})$ is an instance of MSC problem having U as the universal set and $\mathcal{S} = \{S_1, S_2, ..., S_m\}$ where $S_i \subseteq U$, and $\bigcup_{i=1}^{m} S_i = U$. We construct an instance $I_t = (\mathcal{I}, \mathcal{T})$ of the T-COVER problem as follows. Suppose the universal set U corresponds to the set of infected nodes $\mathcal{I}$. The intuition behind this mapping is that with I_s we want to cover each element in U and accordingly we aim to cover all the infected nodes in $\mathcal{I}$ of I_t. Each $S_i \in \mathcal{S}$ is mapped to a time instance $T_i \in \mathcal{T}$ as selection of sets in I_s corresponds to selecting the time instances in I_t when a patch is to be broadcasted. This reduction can be easily done in polynomial time.

We now show that I_s has a MSC iff I_t has a T-COVER. Suppose that I_s has a MSC $C = \{S_{i_1}, ..., S_{i_k}\}$, where $k \geq 1$. Since each $S_{i_j} \in C$ where $j = 1, ..., k$, has a corresponding $T_{i_j} \in \mathcal{T}$ and $\bigcup_{S_{i_j} \in C} S_{i_j} = U$, therefore $\bigcup_{(i,j)|S_{i_j} \in C} T_{i_j} = U = \mathcal{I}$. Thus, $\{T_{i_1}, ..., T_{i_k}\}$, where $k \geq 1$ is a T-COVER of I_t. Conversely, suppose that $W = \{T_{i_1}, ..., T_{i_k}\}$, where $k \geq 1$, is a T-COVER of I_t. Similarly we can prove that for each $T_{i_j} \in W$, we have a corresponding $S_{i_j} \in \mathcal{S}$ and thus $\bigcup_{(i,j)|T_{i_j} \in W} S_{i_j} = U$. Therefore, we conclude that $\{S_{i_1}, ..., S_{i_k}\}$, where $k \geq 1$, is a MSC of I_s. ■

Algorithm 1 Greedy Approximation (ORACLE)

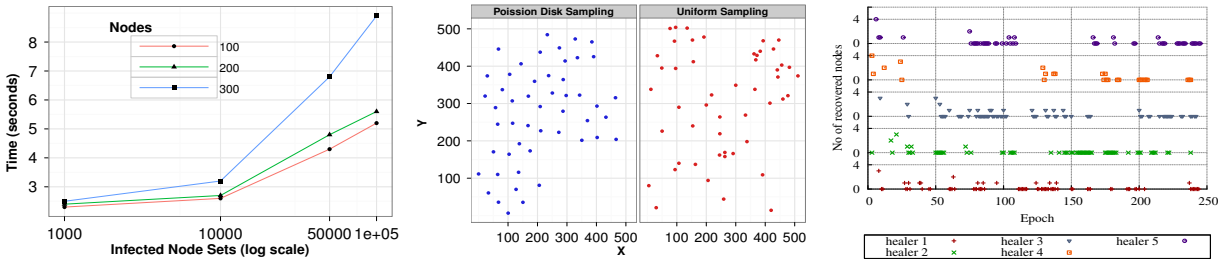
Input Let $\mathcal{I}$ be the list of all infected nodes, S_i be the set of infected nodes seen at each time i, w_i be the list of costs associated with each arrival at i
Initially:
R is the set of elements that are not covered as yet
C is the set of covered elements
w is the weight vector
$R = \mathcal{I}$ and $C = \phi$
repeat
let S_i be the set that minimizes $\frac{w_i}{|S_i \cap R|}$
$C = C \cup \{S_i\}$
$R = R - S_i$
until $R = \phi$
return C

According to the above theorem, we can employ any heuristic that solves the set cover problem to solve the T-COVER problem. Algorithm 1 gives a greedy approximation for the T-COVER. The algorithm takes as input the arrival times of the infected nodes. Here, S_i is the set of infected nodes seen at any one time instant and we equate the weight vector w_i to the time of arrival – cost of healing nodes at a later time is higher because it introduces delay. The main loop iterates for $O(n)$ time, where $|\mathcal{I}| = n$. The minimum W can be found in $O(log\ m)$ time, using a priority heap, where there are m sets in a set cover instance giving us a total time of $O(nlog(m))$. Fig. 5(a) shows that even in the presence of hundreds of thousands of node sets, we are able to compute the optimal solution in under 8 seconds.

C. Effective Healer Placement

Since the healers are static, the healer placement has an impact on our defense protocols and thus their coverage area depends on their placement strategy. Our simulations showed that a naive placement using uniform random distribution resulted in a scenario where many healers ended up covering the same region thereby leaving a lot of uncovered area. Therefore, what we need is a type of a constraint that rejects certain configurations that place healers very close to each other. This problem can be directly reduced to a problem from the field of computer vision which involves producing sampling patterns with a blue noise Fourier spectrum.

Formally, the problem can be defined as the limit of a uniform sampling process with a minimum-distance rejection criterion. Successive points are independently drawn from the uniform distribution $[0, 1]$. If a point is at a distance of at least R from all points in the set of accepted points, it is added to



(a) (b) (c)

Fig. 5. **(a) Oracle Performance:** The algorithm terminates in less than 8 seconds even in long simulation scenarios. **(b) Poisson Disk Sampling:** Healers are no longer placed very close to each other thereby increasing their coverage of the simulation area. **(c) Motivating Backoff:** Most consecutive patches do not heal infected nodes indicating that it is better to backoff after a patch delivery.

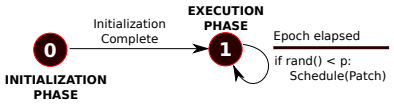


Fig. 6. **State Machine of a Randomized Healer**

that set. Otherwise, it is rejected. The choice of R controls the minimum allowable distance between points. This procedure called *Poisson Disk Sampling* [31] has been actively studied and many efficient algorithms exist. Due to space constraints, we do not discuss this algorithm further but refer the reader to the linear algorithm outlined by [31]. We adapted this algorithm by setting $R = 2r$, where r is the range of our each healer. Fig. 5(b) clearly highlights the merits of using this specific sampling process - healers are no longer close to each other and hence cover more of the simulation area.

D. Family of Randomized Healers

We first present a heuristic where a healer randomly decides at what time within an epoch to send a patch. Note that a healer will decide to send a patch regardless of the number of nodes in its vicinity. Fig. 6 depicts the state machine of the randomized healer (RH). It contains two states, an *initialization phase* where an *epoch timer* is started and an *execution phase* where the healer prepares to deliver a patch. The *epoch timer* fires a callback function that has two responsabilities: (i) pick a random time from the interval $[0, \tau]$, where τ is the *epoch length* that is used to schedule a broadcast, called the *patch timer* and (ii) re-schedule the *epoch timer* to be fired for the next epoch. τ depends on the range of the healer and velocity of the mobile node. When the *patch timer* expires, the healer broadcasts a patch with a probability p, we call it the *patch deployment probability*.

Algorithm 2 Randomized Healers (RH)

Input Epoch length τ and patch deployment probability p
  Initially:
    start $epoch\_timer(\tau)$
  Upon the expiration of $epoch\_timer$:
    select a duration t randomly from $(0, \tau)$
    start $patch\_timer(t)$
    start $epoch\_timer(\tau)$
  Upon the expiration of $patch\_timer$:
    Broadcast a patch with probability p

Algorithm 2 outlines the pseudo-code for the randomized healer. Varying p will generate a family of randomized healers. On one hand, setting $p = 1$ ($RH_{(p=1)}$) makes the healer broadcast a patch at every epoch and thus attempts to minimize the time it takes to heal the system. However, notice that the number of patches delivered would be equal to $\frac{D_{sim}}{\tau}$, where D_{sim} is the simulation duration. On the other hand, setting $p < 1$ makes the healer broadcast a patch only during certain epochs. The time taken to heal the system is inversely proportional to p whereas the number of patches delivered is directly proportional to it.

Algorithm 3 Profile healers w/o (PH) and with (PHB) backoff

Input Epoch length τ, observation time $T(> 1)$ and maximum backoff η
  Initially:
    $t \leftarrow 0$, $\Delta \leftarrow 1$, $state \leftarrow$ LEARNING, $next\_epoch\_time \leftarrow 0$
    $backoff\_enabled \leftarrow$ **true** if $\eta > 1$
    Start $sensing\_timer(\Delta)$
  Upon the expiration of $sensing\_timer$:
    $t \leftarrow t + \Delta$
    if $state =$ LEARNING **then**
      if $t < T$ **then**
        Record $num\_of\_neighbors$ in proximity
      else
        Estimate $threshold$ from the recorded $num\_of\_neighbors$ at each Δ
        $state \leftarrow$ EXECUTION
        $next\_epoch\_time \leftarrow t + \tau$
      Start $sensing\_timer(\Delta)$
    else
      if current $num\_of\_neighbors > threshold$ **then**
        Broadcast a patch
        if $backoff\_enabled$ = **true then**
          Randomly select κ between $(0, \eta)$
          Start $epoch\_timer(next\_epoch\_time - t + \kappa \times \tau)$
        else
          Start $epoch\_timer(next\_epoch\_time - t)$
      else
        Start $sensing\_timer(\Delta)$
  Upon the expiration of $epoch\_timer$:
    if $backoff\_enabled$ = **true then**
      $t \leftarrow get\_current\_time()$
    else
      $t \leftarrow next\_epoch\_time$
    $next\_epoch\_time \leftarrow t + \tau$
    Start $sensing\_timer(\Delta)$

E. Family of Profile Healers

One limitation of the RH approach is that healers may send more patches than needed since they decide to send patches regardless of how many infected nodes are present in their

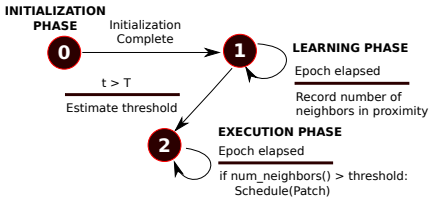


Fig. 7. **State Machine of a Profile Healer**

proximity. We propose a new approach, PH, where a healer attempts to learn the arrival distribution of nodes and subsequently determine whether or not it is cost effective to deliver a patch. The decision is made based on a threshold that captures the number of nodes in its vicinity. Each healer can exist in one of three states as depicted in Fig. 7 - an *initialization phase* which sets up all the relevant timers, a *learning phase* where the healer passively records the number of neighbors it is observing during each epoch, and an *execution phase* where the healer utilizes information that it learnt during the previous phase to decide whether or not to deliver a patch.

The goal of *learning phase* is to learn the distribution of node arrivals specific to a healer's locality for a certain *observation time* T which is a multiple of τ. Specifically, the goal is to learn a threshold of nodes that will determine whether the healer should send a patch or not. We use two metrics: (i) $MSD = Mean + 1.5 \times Standard\ Deviation$ and (ii) $M = Median$. MSD is well-known for normal distributions and makes the healer broadcast a patch only if the number of neighbors exceeds its estimate of the 95^{th} percentile whereas the second metric considers only the median. M is very robust to outliers - it handles cases where a healer observes a burst of infected nodes during an epoch. Algorithm 3 describes this healer in detail (in this case, we use $\eta = 1$).

During our simulations, we observed that relying solely on a *threshold* was leading to a wastage of patches - due to the low contact rate we observed in §III-B. Consider Fig. 5(c) which depicts the healing sequence of a set of five healers during the epochs of one simulation run. Points situated at 0 indicate that the healer deployed a patch as the number of neighbors was above the threshold but the patch *did not* heal any infected nodes. Any other number indicates the number of infected nodes healed with that patch. Observe that most patches are going to waste, *i.e.*, they are not healing any nodes. In the worst case, it takes at least $\frac{healer\ range}{node\ velocity}$ seconds for a node to go out of range of a healer. Therefore, for shorter epochs, consecutive patches are delivered to the same set of nodes. We address this issue by introducing a *random backoff*, i.e., once a patch has been broadcast, the healer selects a random backoff delay κ from the interval $(0, \eta)$, where η is the maximum backoff in epochs, and skips that many epochs. Algorithm 3 also describes the backoff algorithm in detail (in this case, we use $\eta > 1$). We refer to this algorithm as PHB.

We also propose an optimization where the algorithm dynamically estimates the decision threshold, every T epochs, and uses the newly estimated threshold in the next T epochs. We refer to this algorithm as D-PHB.

V. Defense Evaluation

In this section, we describe our evaluation methodology and present the performance of the various healer based defense mechanisms outlined in §IV.

A. Evaluation Methodology

To evaluate the performance of the two families of healers, we simulate these healers using the ns-3 [27] network simulator using a network containing 300 nodes. We perform two different sets of experiments, one with nodes having RWP and the other with TLW as their mobility model. We assume that the range of each healer is 20 meters and the *epoch length*, τ, is 30 seconds (so that each node stays within the range of a healer for one epoch length on an average before leaving the coverage area of the healer). In addition, 10% of the population is assumed to be initially infected to enable bootstrapping the system. We can technically start with one infected node (which was our initial attempt), but we observed that this only delays infection spread and increases the chance that infection will disappear. Healers are placed in the system using the strategy outlined in §IV-C and are activated when the fraction of infected nodes exceeds 70% of the total population to give the system sufficient time to warm-up. We note that 70% is one possible worst case scenario and projects the capabilties of the healer. In real-world scenarios, this value depends on how fast one can setup healers during an epidemic outbreak. Once the healers are activated, they follow the protocols outlined in §IV-D and §IV-E. All results are averaged across 10 runs of each experiment, to obtain statistically significant results, by varying the *seed* of a pseudo-random number generator. To measure the performance of each protocol, we define:

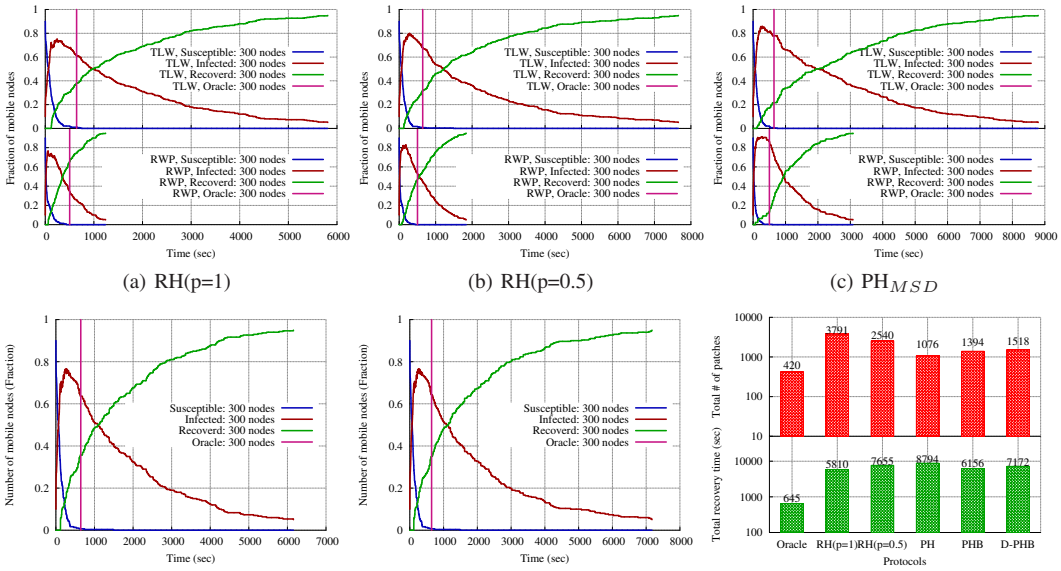
- *Total recovery time*: It represents the amount of simulation time required by the set of healers to recover at least 95% of the nodes in the system.
- *Total number of patches*: It represents the count of patches deployed by the set of healers to heal the system such that at least 95% of the total number of nodes are recovered.

Note that we chose 95% to account for scenarios similar to the *rare block problem* [32] in p2p networks - we observed the presence of infected nodes that take exceedingly long time to enter the range of healers because they are wandering along the edge of the field and hence prolong our simulation.

B. Results for Family of Randomized Healers

Fig. 8(a)-(e) shows the temporal view of infection propagation and the recovery of the system for RH and PH families using RWP and TLW. The graphs show that regardless of the protocol, the required recovery time is always smaller in case of RWP than TLW which is due to RWP's higher contact rate.

Fig. 8(a) shows the required recovery time for randomized healers with $p = 1$, *i.e.*, $RH_{(p=1)}$. The upper graph is for TLW and the lower one is for RWP. Additionally, we also point out the recovery time required by ORACLE using a vertical line. In case of RWP, ORACLE requires 587 seconds to heal the system whereas $RH_{(p=1)}$ requires almost double this time, *i.e.*,



(a) RH(p=1) (b) RH(p=0.5) (c) PH$_{MSD}$

(d) PHB$_M$ with $\eta = 2$ (e) D-PHB$_M$ with $\eta = 2$ (f) Summary

Fig. 8. **Comparison of Healer Families**: (a)-(e) Infection propagation and recovery with the values obtained using our ORACLE shown as the vertical lines, (f) Summary of the performances of healer families for TLW

1,241 seconds. In TLW, $RH_{(p=1)}$ requires about eight times the optimal recovery time. We also note that the recovery time required by $RH_{(p=1)}$ is the minimum time that we can achieve using healers that do not depend on system feedback (*e.g.*, estimating the arrival distribution of nodes).

Fig. 8(b) shows the results for $RH_{(p=0.5)}$, i.e., each healer deploys a patch per epoch with a probability $p = 0.5$. It is expected that $RH_{(p=0.5)}$ requires more time than $RH_{(p=1)}$ to heal the system since now the healers skip some epochs. In comparsion with the recovery time required by $RH_{(p=1)}$, $RH_{(p=0.5)}$ shows 48% increase in case of RWP and 31% increase in case of TLW.

C. Results for Family of Profile Healers

Let X_{MSD} and X_M represent a profile-based healer X that utilizes MSD($= Mean + 1.5 \times Standard\ Deviation$) and M($= Median$) as its threshold, respectively. Fig. 8(c) shows the performance of PH for the RWP and TLW mobility models. PH$_{MSD}$ requires more time to heal the system in comparison with the other two RH healers. Since we are more interested in the human-mimicking mobility model, we evaluate PHB and D-PHB for only TLW in Fig. 8(d) and Fig. 8(e), respectively. Due to space limitation, we present the performance of PHB and D-PHB with maximum backoff $\eta = 2$ and M as the threshold value in Fig. 8(d)- 8(e). When we compare PH$_{MSD}$, PHB$_M$, and D-PHB$_M$ using the TLW mobility model, PHB$_{MSD}$ outperforms the other two in terms of total recovery time.

To measure the impact of different maximum backoff values on the PHB$_{MSD}$ and PHB$_M$, we varied the maximum backoff from 2 epochs to 16 epochs. Fig. 9 shows the results of this experiment. We also include the results of $RH_{(p=1)}$ as a baseline of the performance. We use two Y-Axes for this graph: the left one for the total number of patches and the right one for the total recovery time. Each point is the average of 10 different runs of the simulation and is plotted along its 95% confidence intervals. With the increase in maximum backoff values, the total recovery time is increasing rapidly in case of PHB$_{MSD}$ in comparison with PHB$_M$. On the other side, the total of number of patches is decreasing rapidly for PHB$_{MSD}$ in comparison with PHB$_M$. We conjecture that if the recovery time is to be optimized, then PHB$_M$ is a better solution; but if the energy of the healers is to be optimized, then the PHB$_{MSD}$ is a better choice. However, the downside of PHB is its large observation time. D-PHB is a solution to this downside of PHB. We also include the performance of D-PHB$_M$ in Fig. 9. The results demonstrate that D-PHB$_M$ performs as good as PHB$_M$ in terms of both the metrics. So if the large observation time is unacceptable, D-PHB$_M$ heals the system as fast as PHB$_M$ and does not require any observation time.

Summary: Fig. 8(f) summarizes results for TLW for both metrics obtained by each of the healers. Observe that $RH_{(p=1)}$ outperforms the others in terms of the total recovery time. However, in order to achieve the fastest recovery, $RH_{(p=1)}$ has to deploy the maximum amount of patches. In terms of the number of patches, PH$_{MSD}$ requires the least number of patches but at the cost of a large recovery time. However, PHB$_M$ performs best since it requires only 6% more recovery time in comparison to $RH_{(p=1)}$ and only 30% more patches than that of PH$_{MSD}$.

Our results show that each of the schemes has advantages and disadvantages. Randomized healers offer the immediate advantage that they do not rely on system feedback nor do they have to estimate node arrival distributions. They would be beneficial in a time-constrained system as randomized healers

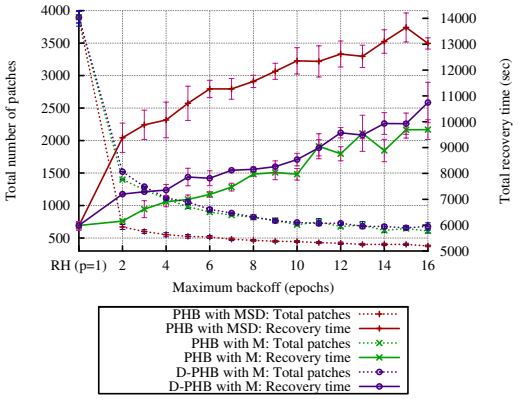


Fig. 9. **Effect of varying maximum backoff**: D-PHB$_M$ performs as good as PHB$_M$ in terms of both the metrics with the added advantage that it does not have an observation time.

are fast at recovering an infected system. However, they result in using 3.5x more patches than the profile healers. On the other hand, profile healers offer intelligent decision making thereby saving energy in the form of utilizing less number of patches and would benefit the most in an energy-constrained environment. However, they result in taking 1.5x more time to recover in comparison to randomized healers.

VI. Related Work

There has been some work in controlling the spread of the worm inside a wireless network [7, 33, 34]. Williamson *et al.* [33] present a technique to limit the rate of connections to "new" machines that is effective at both slowing and halting virus propagation without affecting normal traffic. Their work is based on heuristics and simulations which consider a static choice of reduced communication rate. Wong *et al.* [34] present a technique that relies on limiting the contact rate of worm traffic. Specifically, they investigate rate control at individual end hosts and at the edge and backbone routers, for both random propagation and local-preferential worms. They show that both host and edge-router based rate control result in a slowdown that is linear to the number of hosts implementing the rate limiting filter. Our work focuses on modeling infection dynamics as a function of mobility models and introduces a suite of defense protocols to thwart the epidemic spread.

VII. Conclusion

Mobile malware present an emerging problem that threatens smartphones which are growing significantly in recent days. In this paper, we considered realistic mobility patterns to model proximity dependent malware and compared them against de facto models like random waypoint mobility model. We presented several defense mechanisms that allow tuning parameters to control the dimensions of optimization – either time to recovery or energy utilized. The extensive evaluation of all our defense mechanisms shows that randomized healers would be more effective in a time constrained environment whereas profile healers would benefit the most in an energy constrained environment.

References

[1] "McAfee Threats Report: 3rd Quarter 2011," http://goo.gl/jIQPJ.
[2] "Juniper Mobile Threats Report 2010-11," http://goo.gl/v3yFg.
[3] D. Moore *et al.*, "Inside the Slammer worm," *IEEE SnP*, 2003.
[4] C. Zou *et al.*, "Code Red Worm Propagation Modeling and Analysis," in *Procs. of CCS*, 2002.
[5] A. Wagner *et al.*, "Experiences with worm simulations," in *Rapid Malcode*, 2003.
[6] J. Kephart and S. White, "Directed-graph epidemiological models of computer viruses," *Computation*, p. 71, 1992.
[7] S. Sellke *et al.*, "Modeling and Automated Containment of worms," in *Procs. of DSN 2005*.
[8] A. Khelil *et al.*, "Epidemic model for information diffusion in manets," in *ACM MSWiM*, 2002.
[9] A. Bose *et al.*, "Behavioral detection of malware on mobile handsets," in *Procs. of MSAS*, 2008.
[10] C. Fleizach *et al.*, "Can you infect me now?: malware propagation in mobile phone networks," in *Recurring Malcode*, 2007.
[11] A. Bose and K. Shin, "On mobile viruses exploiting messaging and bluetooth services," in *Securecomm, 2006*, 2006.
[12] G. Zyba *et al.*, "Defending mobile phones from proximity malware," in *Infocom*, 2009.
[13] R. Potharaju and C. Nita-Rotaru, "Pandora: A platform for worm simulations in mobile ad-hoc networks," *ACM MCCR*, 2011.
[14] I. Rhee *et al.*, "On the levy-walk nature of human mobility," *IEEE/ACM TON*, 2011.
[15] M. Khouzani *et al.*, "Optimal quarantining of wireless malware through power control," in *ITA*, 2009.
[16] ——, "Dispatch then stop: Optimal dissemination of security patches in mobile wireless networks," in *CDC*, 2010.
[17] J. Broch *et al.*, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Procs. of ACM MCN*, 1998.
[18] C. Chiang and M. Gerla, "On-demand multicast in mobile wireless networks," in *Procs. of Network Protocols*, 2002.
[19] J. Garcia-Luna-Aceves and M. Spohn, "Source-tree routing in wireless networks," 1999.
[20] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile computing*, pp. 153–181, 1996.
[21] J. Boleng, "Normalizing mobility characteristics and enabling adaptive protocols for Adhoc networks," in *Procs. of MAN*, 2001.
[22] T. Camp *et al.*, "A survey of mobility models for ad hoc network research," *WCMC*, 2002.
[23] V. Capasso and G. Serio, "A generalization of the kermack-mckendrick deterministic epidemic model," *Math. Biosci*, 1978.
[24] C. Huang *et al.*, "Influence of local information on social simulations in small-world network models," *Journal of Artificial Societies and Social Simulation*, vol. 8, no. 4, 2005.
[25] R. Potharaju, "Infection quarantining for wireless networks using power control," in *DSN Student Forum*, 2010.
[26] C. de Waal and M. Gerharz, "Bonnmotion: A mobility scenario generation and analysis tool," *CSG*, 2003.
[27] "The ns-3 Network Simulator," http://goo.gl/Lh1Fw.
[28] C. Bettstetter *et al.*, "The spatial node distribution of the random waypoint mobility model," in *WMAN*, 2002.
[29] G. Resta and P. Santi, "Analysis of node spatial distribution of Random Waypoint Model for Adhoc networks," in *WMC*, 2002.
[30] E. Hyytiä and J. Virtamo, "Random waypoint mobility model in cellular networks," *Wireless Networks*, 2007.
[31] R. Bridson, "Fast poisson disk sampling in arbitrary dimensions," in *ACM SIGGRAPH*, vol. 2007, 2007.
[32] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," in *Infocom*, 2005.
[33] M. Williamson, "Throttling viruses: Restricting propagation to defeat malicious mobile code," in *CSA*, 2002.
[34] C. Wong *et al.*, "Dynamic quarantine of Internet worms," 2004.