

# Worm Origin Identification Using Random Moonwalks

Yinglian Xie    Vyas Sekar    David A. Maltz    Michael K. Reiter    Hui Zhang  
Carnegie Mellon University\*

## Abstract

*We propose a novel technique that can determine both the host responsible for originating a propagating worm attack and the set of attack flows that make up the initial stages of the attack tree via which the worm infected successive generations of victims. We argue that knowledge of both is important for combating worms: knowledge of the origin supports law enforcement, and knowledge of the causal flows that advance the attack supports diagnosis of how network defenses were breached. Our technique exploits the “wide tree” shape of a worm propagation emanating from the source by performing random “moonwalks” backward in time along paths of flows. Correlating the repeated walks reveals the initial causal flows, thereby aiding in identifying the source. Using analysis, simulation, and experiments with real world traces, we show how the technique works against both today’s fast propagating worms and stealthy worms that attempt to hide their attack flows among background traffic.*

## 1 Introduction

In all propagating worms, epidemic spreading attacks, and other types of attacks that utilize compromised computers to launch attack traffic, the overwhelming majority of the attack traffic originates from victims of the attack, as opposed to the true source of the attack. This affords the attacker a great degree of anonymity, and to date there is little automated support for identifying the location (computer or network) from which such an attack is launched. Similarly, when an intranet succumbs to such an attack, there is little automated help to determine the internal computer that was compromised first.

In [21], we have argued that it is important for the network to support automatic forensic analysis abilities *after*

---

\*This research was supported in part by National Science Foundation grant number CNS-0433540 and ANI-0331653 and U.S. Army Research Office contract number DAAD19-02-1-0389. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of NSF, ARO, Carnegie Mellon University, or the U.S. Government or any of its agencies.

an attack has happened. We have proposed a general Dragnet [8] framework to support network auditing and forensic capabilities. In this paper, we investigate the specific problem of crafting an algorithm that determines the origin of epidemic spreading attacks such as Internet worms. Our goal is not only to identify the “patient zero” of the epidemic, but also to reconstruct the sequence of events during the initial spread of the attack and identify which communications were the *causal flows* by which one host infected the next. Identifying the causal infection flows allows investigators to study how the attack managed to bypass security barriers intended to stop attacks, such as firewalls between departments in an enterprise’s intranet.

Prior research on worm attacks has largely focused on the detailed study of specific attacks seen in the wild, e.g., analyzing their scanning strategies and the vulnerabilities they exploit in order to develop better signatures for flows that are likely to be worm infection attempts. In contrast, our research takes the extreme opposite approach. We ask a deliberately broad question: is it possible to identify the worm origin without *any* a priori knowledge about the attack?

Our algorithm for detecting worm attack origin is based on the one invariant across all epidemic-style attacks (present and future): for the attack to progress there must be communication among attacker and the associated set of compromised hosts, and the communication flows that cause new hosts to become infected form a causal tree, rooted at the source of the attack. While these flows may be subtle or invisible when observed individually from any single host, the tree structure will potentially stand out when viewed collectively. By creating algorithms that work by identifying the overall structure of an attack’s propagation, our approach can be agnostic to attack signatures or scanning rates and potentially be applicable to all worm attacks.

The algorithmic challenge is daunting even if, as assumed in this paper, the complete graph of host communication is available. Our goal is an algorithm that can find large tree-structured subgraphs, and thus the root of such trees, of the host contact graph defined in Section 3, where the edges are all the flows that happened in the network. We know of no tractable algorithm for finding such sub-

graphs in very large graphs. We are drawn to a formulation based on finding tree structures as, in addition to finding fast-spreading worms, we also want to find *slow-spreading* worms, where each infected host makes infection attempts at a rate significantly below the rate of normal traffic. Given its exponential growth pattern, a slow worm merely requires a few extra generations to achieve the same spread as a fast worm, while being significantly harder to catch as it blends in with normal traffic.

This paper presents the *random moonwalk* algorithm that can find the origin and the initial propagation paths of a worm attack, either within an intranet or on the Internet as a whole, by performing post-mortem analysis on the traffic records logged by the networks. The algorithm works by repeatedly sampling paths on the host communication graph with random walks. Each walk randomly traverses the edges of the graph *backwards* in time, and hence the name *random moonwalk*.

The algorithm depends only on the assumption that worm propagation occurs in a tree-like structure from its origin, where an infection flow from one computer (the “parent”) to its victim (the “child”) forms a directed “edge” in this tree. We show that in the presence of a large-tree structured subgraph, these walks tend to be directed towards the root of the tree so that correlating many walks reveals the structure of the initial levels of the tree. We demonstrate through analysis, simulation, and experiments on real world traces that this approach can be highly effective in locating the origin of an attack, without the use of attack signatures for detection. We evaluate the algorithm against a variety of background traffic patterns and worm spreading-rates, showing its effectiveness even against slow-spreading worms.

The primary contribution of this paper is an algorithmic solution to identify the epidemic attack source and the initial causal flows. By exploiting attack invariants such as the globally visible attack structure, our algorithm is agnostic to attack signatures, port numbers used, or specific software vulnerabilities exploited. Thus it has the potential to be robust to future stealthy attacks that have not been seen in networks today.

## 2 Related Work

To our knowledge, we are not aware of any previous work that can automatically pinpoint the origin of an epidemic attack or the initial causal infection events.

Our algorithm assumes that attack flows do not use spoofed source IP addresses, since in the types of attacks we consider here, attack packets are rarely, if ever, spoofed. The overwhelming majority of attack traffic involved in the propagation is initiated by victims instead of the original attacker, so using spoofed addresses would only decrease

the number of successful attacks<sup>1</sup> without providing extra anonymity to the attacker.

If attackers do begin to use spoofed addresses, then traceback techniques [2, 6, 15, 19, 23] could be used to determine the true source of each flow sampled by our algorithm. Traceback alone, however, is not sufficient to track worms to their origin, as traceback determines only the true source of the packets received by a destination. In an epidemic attack, the source of these packets is almost never the origin of the attack, but just one of the many infected victims. Some method is still needed to find the hosts higher up in the causal tree.

Other work on traffic causality analysis has mostly focused on detecting stepping stones, which is suggested [22] as a potential solution for worm origin identification together with IP traceback. Just as we discussed that IP traceback cannot be used to trace the origin of epidemic attacks, stepping stone techniques are not suitable for our objectives either.

There have been in general two categories of approaches for detecting stepping stones. Content-based techniques [24] require expensive packet payload analysis, but cannot track down flows from polymorphic worms or worms that encrypt payloads. The other class of approaches [7, 29] focus on correlating packet-level characteristics (e.g., inter-packet timings) to detect if multiple interactive connections are part of a single attack session. However, using fine-grained packet timing characteristics for establishing causality does not work for worm attacks which typically do not use interactive sessions. Even in the context of detecting causality of interactive flows, such techniques still remain an active area of research especially with respect to the robustness of such timing correlations [4, 26]. In contrast, our work ignores packet-level characteristics and attack signatures, but instead focuses on establishing causal relationships between flows by exploiting the globally visible structure of attacks. Thus our algorithm can potentially be agnostic to specific attack contents, attack packet sizes, or port numbers used.

While our work does not depend on the generation of worm signatures, our approach is complementary to these efforts [12, 13] as well as other efforts in detecting the existence of attacks [10, 11, 16, 28] and traffic anomalies [1]. Finally, our method for correlating random walks is inspired by link analysis [14], where the authors infer correlations among social network entities from their activity patterns.

---

<sup>1</sup>For example, spoofed packets are useless for propagating an infection over TCP-based communications, since the TCP handshake cannot complete, and spoofing addresses for UDP-based attacks in the presence of egress filters [9] results in the attack flows being discarded.

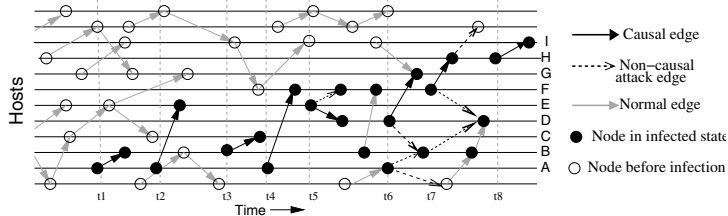


Figure 1: Example of host contact graph showing the communication between hosts. Attack edges are shown as arrows in black (both solid and dashed). Filled nodes correspond to hosts in an infected state.

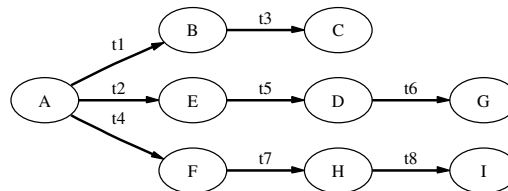


Figure 2: Example showing the causal tree, which contain causal edges with timestamps from the host contact graph.

### 3 Problem Formulation

We model the network communication between end-hosts using a directed *host contact graph*  $G = \langle V, E \rangle$ . The nodes of the graph  $V = H \times T$ , where  $H$  is the set of all hosts in the network and  $T$  is time. Each directed edge represents a network *flow* between two end hosts at a certain time, where the flow has a finite duration, and involves transfer of one or more packets. We represent each edge by a tuple  $e = \langle u, v, t^s, t^e \rangle$  where  $u \in H$  is the host that initiates the communication (the source of the flow),  $v \in H$  is the host that receives the communication (the destination of the flow), and  $t^s, t^e \in T$  are the start and end times of the flow. Edge  $e$  is thus from node  $(u, t^s) \in V$  to node  $(v, t^e) \in V$ . We have found that including time in the model is important, as a single host  $h \in H$  that becomes infected during an attack behaves differently before the time it is infected than it does afterwards.

Figure 1 shows the host contact graph of a hypothetical network undergoing an attack. Time advances left to right. Each node (marked as a circle) in the graph corresponds to the state of a host at a certain time. The nodes on the same horizontal line show how the state of one host changes over time, and the nodes on the same vertical line represent the states of different hosts at the same time.

Each directed edge in Figure 1 represents a network flow. If a flow does not carry an infectious payload, we call that edge a *normal edge*. We define an edge as an *attack edge* (highlighted in the figure as either dashed or solid arrows) if it corresponds to a flow that carries attack traffic, whether or not the flow is successful in infecting the destination host. While a worm attack may induce a large number of attack flows in the network, only a few flows actually advance the attack by successfully infecting a new host. We define an edge as a *causal edge* (highlighted as a solid arrow) if it corresponds to a flow that actually infects its destination. For example, at time  $t_6$ , host D has attack edges to both hosts G and B. However, only the edge from D to G is a causal edge because G is infected by this contact, whereas B was infected earlier before time  $t_2$ .

The *causal tree* formalizes the concept of epidemic at-

tack spread. The causal tree is formed by extracting the causal edges from the host contact graph and projecting the edges along the time axis. To be consistent with the notion of time in the host contact graph, we consider causal edges occurring earlier in time as edges in the higher levels of the causal tree. Figure 2 shows the causal tree for the attack in Figure 1, with each edge annotated with a timestamp. The edge with timestamp  $t_1$  from the worm origin A is thus at the highest level of the tree.

Given a host contact graph, the goal of our algorithm is to *identify a set of edges that, with high probability, are edges from the top level(s) (i.e., initial in time) of the causal tree*. Among the hosts listed as the sources of these edges will be the origin of the attack (or the host at which the attack first entered the intranet). It is critical that the technique have a reasonably low false-negative rate, so that the returned set contains at least one top level causal edge that identifies the attack origin. It is desirable that the technique have a low false-positive rate, so that the returned set does not include many normal edges, attack edges that do not infect the destination, or even causal edges that occur lower in the causal tree, since the sources of these edges are less likely to be the true origin of the attack.

### 4 The Random Moonwalk Algorithm

Our algorithm consists of repeatedly sampling paths from the host contact graph and then correlating these samples. The edges that occur most frequently among the samples are selected as the edges most likely to be causal edges from levels higher up in the causal tree. The first key to the technique is that we do not sample individual edges — rather, each sample is a contiguous path of edges in the graph. The second key is that we create the path by starting at a randomly chosen edge, and then *walking backwards in time* along the graph, randomly choosing among potential predecessor edges at each step in the moonwalk.

The sampling process is controlled by three parameters:  $W$  - the number of walks (i.e., samples) performed,  $d$  - the maximum length of the path traversed by a single walk, and  $\Delta t$  - the sampling window size defined as

the maximum time allowed between two consecutive edges in a walk. Each walk starts at an arbitrarily chosen edge  $e_1 = \langle u_1, v_1, t_1^s, t_1^e \rangle$  representing a flow from host  $u_1$  to host  $v_1$ . We then pick a next step backward in time uniformly from the set of edges that arrived at  $u_1$  within the previous  $\Delta t$  seconds. That is, an edge  $e_2 = \langle u_2, v_2, t_2^s, t_2^e \rangle$  such that  $v_2 = u_1$  and  $t_2^e < t_1^s < t_2^e + \Delta t$ . Each walk stops when there is no edge within  $\Delta t$  seconds to continue the path, or the path has traversed the specified maximum number of hops  $d$ .

As the sampling is performed, a count is kept of how many times each edge from the host contact graph is traversed. After  $W$  walks have been performed, the algorithm returns the  $Z$  edges with the highest counts. Here,  $Z$  is a user specified parameter to determine how many edges are to be returned for further investigation. These edges are most likely to be top-level causal edges from the causal tree. As defined and used in this paper, the algorithm operates off-line with the parameters and host contact graph as inputs. As future work, we are investigating on-line versions that may also dynamically tune parameters.

Each random moonwalk made by the algorithm samples a *potential* causal chain of events. Because the walks wander into the past, the edge at step  $i$  (time =  $t_1$ ) in a walk could be potentially caused by the edge at step  $i + 1$  (time =  $t_2$ , where  $t_2 < t_1$ ). Since the walks begin at different randomly chosen edges, an edge that shows up frequently among many walks has the potential to be indirectly responsible for causing a large number edges in the host contact graph. Worm attacks have the property that a small number of edges (those high up in the causal tree) *are* indirectly responsible for causing a large number of edges in the host contact graph (the attack edges lower in the tree). Thus the edges implicated by our sampling algorithm are likely to be those high in the causal tree.

Two factors appear to aid in the convergence of the sampling algorithm, although it remains future work to determine the relative importance of each factor.

First, an infected host generally originates more flows than it receives. If the worm makes attack attempts very rarely this difference may be slight, but sending attack flows increases the rate of outgoing flows without increasing the rate of incoming flows. The result is that there are more edges that can lead a walk to an infected host than there are edges that lead away from it. This tends to concentrate walks towards the root of the tree.

Second, in normal communication patterns today, most hosts are clients that initiate communication with servers, and so are the originators of flows in the host contact graph. Since hosts receive relatively few flows, random moonwalks in a host contact graph without an ongoing worm attack tend to be very short, as many edges have no predecessors within the  $\Delta t$  sampling window. Worms, port

scanning, and peer-to-peer systems are among the few applications that cause hosts to receive flows, and port scanning or peer-to-peer systems tend to lack the tree-structure that cause random moonwalks to concentrate.

## 5 Evaluation Methodology

We evaluate the random moonwalk algorithm using an analytical study, real trace experiments, and simulations, with different models of background traffic and different worm propagation rates. We first present in Section 6 analytical results with a simplified traffic model, showing that the random moonwalk technique has promise, and give analytical estimates on the performance of the algorithm. Section 7 presents experimental results with a large real network trace, to demonstrate the success of the algorithm in discovering the initial causal edges under various attack scenarios including worms propagating at very slow rates. We also discuss how to select the best parameter values for maximum walk length  $d$  and sampling window  $\Delta t$  for an arbitrary network trace. For completeness, we present in Section 8 a set of simulation experiments to show the performance of the algorithm under different background traffic models.

As discussed earlier, the output of the random moonwalk algorithm is a set of the  $Z$  edges that were traversed most frequently during the  $W$  moonwalks. Given the  $Z$  returned edges, we use three performance metrics to evaluate the performance of the algorithm: (1) the detection accuracy in terms of the number of causal edges and attack edges returned, (2) the false positive rate of the set of edges returned, and (3) the number of suspect hosts identified by the algorithm as potential origins of the worm.

As our goal is to identify the initial causal edges whose source is the worm origin, attack edges and even causal edges from lower levels of the causal tree are considered as false positives. In the analytical study, we develop a model for reasoning about the false positive rates associated with finding only the *top-level causal edges*. In real attacks, the notion of *top-level* edges loses meaning, since the assumptions simplifying the notion of time and the unit duration of a flow (made in the analysis) no longer hold. Therefore, in the simulation and real trace studies, we evaluate performance using *detection accuracy* of the number of causal edges among the  $Z$  top frequency edges. We then use experiments to show that the majority of the returned causal edges are from the highest levels of the causal tree, with the worm origin as one of the sources of the edges.

## 6 Analytical Model

In this section, we present an analytical model that explains how well the random moonwalk sampling process works and why. Using the analytical model, we show how we can

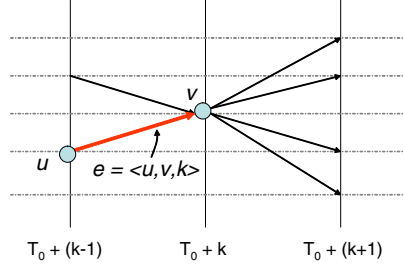


Figure 3: An edge at time  $k$  in the host contact graph.

predict the sampling performance achieved from  $W$  walks with maximum length  $d$  and given  $\Delta t$ .

## 6.1 Assumptions

To enable tractable analysis of the random moonwalk sampling, we make simplifying assumptions about the structure of the host contact graph and the attack. Although our model is an over-simplification of real network traffic, it enables an estimation predicting the performance of the technique and sheds light on the intuition behind the effectiveness of the technique.

First, we assume the host contact graph is known, and it contains  $|E|$  edges and  $|H|$  hosts.

Second, we discretize time into units. We assume every flow has a length of one unit, and each flow starts at the beginning of a unit and finishes before the start of the next unit.

Third, we define the start time of the first attack flow,  $T_0$ , to be the origin of the time axis. Combined with the second assumption, this means that rather than describing both the start and end times of an edge in terms of continuous time variables, we can refer to its “time” as  $k = t^e - T_0$  using just the flow end time  $t^e$ . The first attack edge is then at time  $k = 1$ , and an edge  $e = \langle u, v, t^s, t^e \rangle$  is at time  $k$  if  $t^e = T_0 + k$  (illustrated in Figure 3). In the analysis below, we use  $e^k$  to denote an edge at time  $k$ ,  $e^k = \langle u, v, k \rangle$ . Edges that occurred before  $T_0$  will have negative  $k$  values.

Fourth, we assume a normal host initiates  $B$  concurrent outgoing flows at each time unit. Once a host is infected, it starts malicious scanning by initiating a total of  $A$  outgoing flows at each subsequent time unit. The  $A$  outgoing flows include  $B$  normal flows and  $A - B$  attack flows. Both the normal hosts and the infected hosts randomly select a destination host for every flow. Unlike a normal flow, not every attack flow will go to a valid host address. Suppose only fraction  $r$  of the address space is being used, then among the  $A - B$  concurrent outgoing attack flows,  $R = (A - B) \times r$  will go to existing hosts, while the rest  $A - B - R$  will go to invalid destinations. This results in an infected host initiating a total of  $B + R$  flows to valid destinations each time unit. The rate at which the worm spreads is thus determined

by both  $A$ , the rate of scanning, and  $R$ , the effectiveness of the scans.

Finally, we assume that flows and packets are not lost or blocked, so that flows sent to a valid host are received by that host. This means that the total number of flows sent to valid hosts at time  $k - 1$  will be the total number of flows received at time  $k$ . If the fraction of infected hosts at time  $k - 1$  is given by  $f(k - 1)$ , then each host at time  $k$  will receive an average of  $I(k)$  flows, where

$$I(k) = \underbrace{(B + R) \times f(k - 1)}_{\text{Flows from infected hosts}} + \underbrace{B \times (1 - f(k - 1))}_{\text{Flows from normal hosts}} \quad (1)$$

With the notions introduced above, we can simplify the random moonwalk algorithm described in Section 4. For each walk, once we select an edge  $e_1 = \langle u_1, v_1, k_1 \rangle$  as our current step, we consider an edge  $e_2 = \langle u_2, v_2, k_2 \rangle$  as a candidate next step only if  $v_2 = u_1$  and  $k_2 + 1 = k_1$ , i.e.,  $\Delta t = 1$ .

## 6.2 Edge Probability Distribution

With the above assumptions and notation, we show analytically that the initial causal flows are more likely to be traversed by a random moonwalk, and thus be selected for identifying the ultimate source or entry point of the attack. We do so by estimating  $P(e)$  — the probability of an edge  $e$  being traversed in a random moonwalk on the host contact graph.

We classify edges into two categories based on their destinations. We define an edge  $e_m^k = \langle u, v, k \rangle$  as a *malicious-destination* edge if  $v$  is infected before or at time  $k$ . Otherwise, we define the edge as a *normal-destination* edge denoted as  $e_n^k$ . Since a causal edge will successfully infect the destination host immediately, a causal edge is always a malicious-destination edge. With the two categories of edges, we have the following approximations:

$$P(e_w^k) \approx \begin{cases} \frac{1}{|E|} \left[ 1 + \frac{A}{I(k)} + \frac{(B+R) \times \sum_{i=1}^{d-2} T_{k+i}}{I(k)I(k+1)} \right] & w = m; \\ \frac{1}{|E|} \left[ 1 + \frac{B}{I(k)} + \frac{B \times \sum_{i=1}^{d-2} T_{k+i}}{I(k)I(k+1)} \right] & w = n. \end{cases}$$

where  $T_k = Af(k) + B[1 - f(k)]$ . We present how we derive the above estimates in the Appendix. Based on the above observations, the probability difference between the two categories of edges is estimated as:

$$P(e_m^k) - P(e_n^k) \approx \frac{1}{|E|} \left[ \frac{A - B}{I(k)} + \frac{R \sum_{i=1}^{d-2} T_{k+i}}{I(k)I(k+1)} \right] \quad (2)$$

For fast propagating worms,  $A \gg B$  and  $R > 0$ , so it is clear malicious-destination edges (hence causal edges) have

higher probability of being selected by the random moonwalks than normal-destination edges. The difference between the two probabilities (hence the effectiveness of random moonwalks) increases as the path length  $d$  increases and as the scanning rate  $A$  increases (i.e., the worm is more aggressive).

The analytic model presented in this section makes a worst-case assumption that both normal and attack traffic choose the destination for each flow uniformly from among all possible hosts. Therefore, it cannot predict the performance of the algorithm on worms that send attack flows less frequently than normal flows (i.e., setting  $A < B$  is meaningless). In the sections that follow, we show experimental evidence that the algorithm is effective even for very stealthy worms where infected hosts send attack flows more slowly than the rate at which normal flows are sent.

Interestingly, the effectiveness of the random moonwalk algorithm *increases* as the scan rate to valid hosts  $R$  increases. This means that the *fewer* packets the worm sends to invalid addresses, the *easier* it is to catch, which nicely complements honey-pot techniques that detect worms that send many packets to non-existent destinations.

To estimate how  $P(e)$  distributes as an attack evolves, we need to estimate both  $I(k)$ , the expected number of incoming edges at a host at time  $k$ , and  $f(k)$ , the fraction of infected hosts in the network. The fraction of infected hosts  $f(k)$  can be estimated using a logistic equation [25] that models the growth rate of epidemics. Since an infected host randomly scans the network to propagate the attack, among the total  $R$  concurrent outgoing attack flows to valid hosts,  $R \times [F - f(k - 1)]$  flows will infect vulnerable hosts that have not been infected before, where  $F$  is the fraction of vulnerable hosts in the network. Thus

$$f(k) = \begin{cases} 1/|H| & k = 0 \\ f(k-1) [1 + R \times (F - f(k-1))] & k > 0 \end{cases}$$

Figure 5 shows the growth of the fraction of infected hosts as a fast propagating worm progresses on the host contact graph described by parameters in Figure 4. We observe that as the attack advances, the number of infected hosts grows quickly until all vulnerable hosts are compromised and the attack saturates. This rapid growth results in a non-uniform probability distribution of the edges being traversed.

Figure 6 shows how  $P(e_m^k)$  and  $P(e_n^k)$  change over time in an attack scenario as described in Figure 4 with  $d$  set to 10 hops. The attack starts at time 0 and ends at time 15, so there are no values for  $P(e_m^k)$  shown outside this range. The graph shows that the probability  $P(e)$  is highest for malicious-destination edges at times close to the start of the attack. This occurs because the rapid spread of the worm and its zealous scanning means that for time  $k > 2$ , the majority of the edges received by a host are from infected hosts

(i.e.,  $(B + R) \times f(k - 1) > B \times [1 - f(k - 1)]$  for  $k > 2$ ). This results in almost all walks started at times  $k > 2$  selecting an attack edge as the next step backward. Further, as the total number of infected hosts increases with time,  $I(k)$  increases monotonically in the time interval  $[0, 5]$  (the attack saturates at  $k = 4$ ). Therefore, random moonwalks tend to traverse edges between infected hosts, and converge to the topmost levels of the causal tree. The probability of traversing a normal edge at time  $k$ ,  $P(e_n^k)$ , is a constant until  $k = -5$  at which point it grows until  $k = 2$ , shortly after the attack starts. This growth occurs because walks started at times  $0 < k < 10$  tend to concentrate as they walk backward in time along the attack edges until they walk past the beginning of the attack, at which point they begin diffusing through the normal edges. Thus normal edges received by nodes infected early in the causal tree are sampled more frequently than normal edges that occurred at  $k < -5$ .

Equation 2 and Figure 6 suggest that random moonwalks will be most effective in selecting the malicious-destination edges that occur at the highest levels of the causal tree. Identifying these edges, in particular the  $k = 1$  edges, reveals the origin or entry point of the attack.

### 6.3 False Positives and False Negatives

The output of the random moonwalk process is a set containing the  $Z$  edges with the highest frequency counts after  $W$  walks. From this set, we are particularly interested in finding the  $k = 1$  causal edges, because the source of these edges is the origin of the attack. In this section, we analytically study the effectiveness of our algorithm by calculating the expected false positive and false negative rate for the  $k = 1$  causal edges using the definitions below:

- *false positive rate* is the number of non-causal edges and the number of  $k > 1$  causal edges in the set divided by the total number of non-causal edges; and
- *false negative rate* is the number of  $k = 1$  causal edges not identified divided by the total number of causal edges.

Notice with this definition, we consider failed infection attempts (those scans that reach non-vulnerable hosts), repeated infection attempts (those scans that reach already infected hosts), and even lower level causal flows (those scans that successfully infect hosts at time  $t^e > 1$ ) as false positives, if identified by our algorithm.

The number of times a  $k = 1$  causal edge appears in  $W$  random moonwalks can be represented as a random variable  $\mathbb{X}$  that follows a binomial distribution with  $p = P(e_m^1)$ . For large  $W$ ,  $\mathbb{X}$  can be approximated by a normal distribution [27] with mean  $\mu = p \times W$  and standard deviation  $\sigma = \sqrt{p(1-p)W}$ . To ensure the  $k = 1$  causal edges are included in the output set with a false negative rate of  $\alpha$ , we

Number of Edges $ E $	$4.9 \times 10^7$
Number of Hosts $ H $	$10^5$
Vulnerable fraction $F$	0.1
Valid host space $r$	0.5
Normal rate $B$	2
Infection rate $A$	400
Attack start time	0
Attack stop time	15

Figure 4: The parameters of a host contact graph with a fast propagating worm.

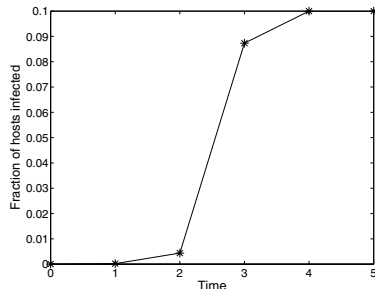


Figure 5: Fraction of infected hosts as an attack advances. The total fraction of vulnerable hosts is 0.1.

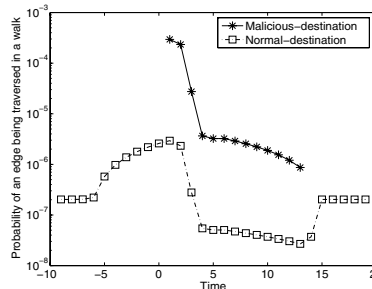


Figure 6: Estimated probability of an edge being traversed in one random moonwalk.

need to select all the edges whose sample frequencies are above a threshold value of  $Z_\alpha$  such that  $Pr(\mathbb{X} < Z_\alpha) = \alpha$ .

Among the selected edges will be the desired  $k = 1$  causal edges and three types of false positives: (1) normal-destination edges, (2) malicious-destination edges with  $k > 1$  (both causal and non-causal edges), and (3)  $k = 1$  malicious-destination, but non-causal edges (i.e., a normal flow sent to a host at  $k = 1$  which was also infected at  $k = 1$ ). The last type of false positives arise because these normal edges have the same probability of being sampled as a  $k = 1$  causal edge. These errors are unavoidable, but false positives from the first two categories can be reduced by increasing  $W$ .

To illustrate the performance of the algorithm, we use the same host contact graph described by Figure 4 where there are in total  $10^4$  causal flows out of the  $4.9 \times 10^7$  flows. Among the 42 malicious-destination edges at  $k = 1$ , 20 are causal edges while the remaining 22 fall under the third category of false positives (i.e., normal edges sent to a host that was infected at  $k = 1$ ); which means that in the ideal case 1 out of 2 edges selected will be causal edges. To estimate the false positives arising from the first two categories, we need to compute the probability of an edge  $e$  with  $P(e) = p'$  having sample frequency  $\mathbb{X}'(e) \geq Z_\alpha$  over the  $W$  random moonwalks, where  $e$  is either a normal-destination edge or a malicious-destination edge with  $k > 1$ . Again,  $\mathbb{X}'(e)$  is a random variable approximated by a normal distribution. With a threshold value of  $Z_\alpha$  used to select edges, suppose  $Pr(\mathbb{X}'(e) \geq Z_\alpha) = \beta$ . Let  $|E(p')|$  be the total number of edges with  $P(e) = p'$ , then  $\beta|E(p')|$  edges will have sample frequencies larger than the threshold  $Z_\alpha$  and be falsely included in the output set.

Figure 7 plots the false negative rate vs. false positive rate for identifying the  $k = 1$  causal edges as the number of walks  $W$  varies using the parameters described in Figure 4. In general, the false positive rates are low even for small false negative rates. With  $10^6$  walks, the false positive rate is  $0.5 \times 10^{-6}$  with a false negative rate of 0.1. This means

that the chance of a non-causal edge or a lower-level causal edge being selected by the technique, when 90% of the  $k = 1$  causal edges are identified, is about 0.5 in a million. The false positive rate drops with increased number of walks, but the rate of decrease slows when the number of walks is larger than  $10^6$ .

We are primarily interested in identifying the worm origin, and the source of every flow returned by the algorithm is a candidate for the origin of the worm. Thus it would be ideal to present to a network administrator a small set of suspect hosts that need to be investigated further. We define the *origin identification false positive rate* as the number of innocent hosts among the sources of the flows selected by the algorithm divided by the total number of hosts minus one (we assume the worm has a single origin). We compute a conservative upper bound by assuming every selected flow returned by the algorithm is from a unique source.

Figure 8 plots the origin identification false positive rate vs. causal edge false negative rate for different numbers of walks. Since there are multiple causal edges from the worm origin, identifying the origin should work well even if there is a slightly higher false negative rate for causal edges. In this example, if we wish to select 70% of the  $k = 1$  causal edges to confirm the attack origin, then after  $10^6$  walks there will be at most 16 candidate hosts for the worm origin from a total of  $10^5$  hosts, greatly reducing the suspect set for further investigation.

## 6.4 Parameter Selection

Understanding the impact of the choice of input parameters  $d$  and  $W$  on the performance of the random moonwalks is important as these parameters determine the amount of sampling effort required. Figure 9 shows the false positive rate for different values of  $d$  (the maximum length of the random moonwalk) and  $W$  (the number of walks) with the false negative rate held constant at 0.1. We observe that longer walks generally result in lower false positive rates. This is also suggested by Equation 2, where the difference

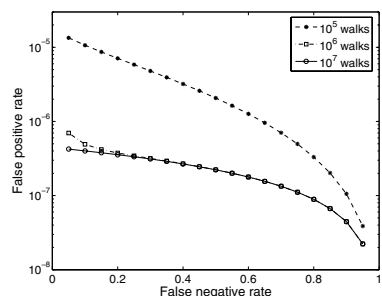


Figure 7: False negative rate vs. false positive rate of finding  $k = 1$  causal edges.

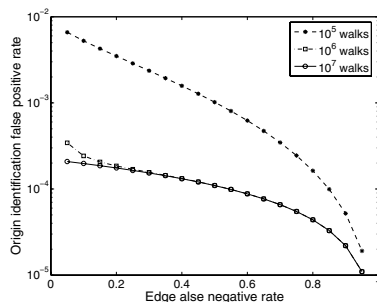


Figure 8: Estimation of the maximum false positive rate of identifying the attack source.

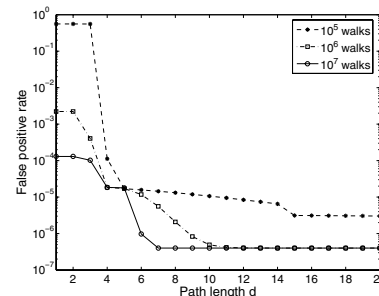


Figure 9: False positive rate of finding  $k = 1$  causal edges vs. maximum path length  $d$ .

between  $P(e_m^k)$  and  $P(e_n^k)$  increases as  $d$  increases. The reason is that when random moonwalks start from lower level edges of the attack tree, they may end before reaching the origin of the attack, increasing the false positive rate. We will further address the impact of parameters  $d$  and the sampling window size  $\Delta t$  on performance using real-world traces in Section 7.4.

## 7 Real Trace Study

In this section, we present our experimental results using real world traces collected from a university network. The objective of the trace based study was to both test the effectiveness of our algorithm using real traffic and to study the performance of the algorithm in different attack scenarios. As our analytical study argues the effectiveness of the algorithm for fast propagating attacks, we focus the real trace study on stealthy attacks that generate low traffic volumes that might escape traditional scanner and super-spreader detection mechanisms.

The traffic trace was collected over a four hour period at the backbone of a class-B university network, where we can observe a significant fraction of the intra-campus network traffic. Each record in the trace corresponds to a directional flow between two hosts with timestamps. We excluded flow records between campus hosts and non-campus hosts to study the performance of our technique on worm propagation inside an intranet. The resulting trace has about 1.4 million flows involving 8040 campus hosts.

With the four hour trace serving as real-world background traffic, we add flow records to the trace that represent worm-like traffic with varying scanning rates. We vary the fraction of vulnerable hosts  $F$ , by randomly selecting the desired fraction of hosts from the set of 8040 total internal hosts. For the following experiments, except Section 7.7, we choose  $F = 0.1$ . Each worm outbreak starts roughly 2800 seconds into the trace, and lasts for 8000 seconds. Once a host is infected, it generates one attack flow every  $t$  seconds to a randomly selected destination from

among the 8040 hosts. In the real trace, 90% of the hosts send fewer than one flow every 20 seconds. To describe how aggressive a worm is, we define the *normalized worm rate* as the ratio of the rate an infected host sends attack flows to the 90 percentile of the normal connection rate (e.g., a worm sending one flow per 20 second has a normalized worm rate of 1, and a worm sending one flow every 200 seconds has a normalized rate of 0.1). Figure 10 lists the characteristics of the worms we introduced to the real world trace. We use “Trace- $x$ ” to refer a trace with worm rate of one attack flow per  $x$  seconds.

We introduce two additional metrics to compare the performance across worms of different scanning rates. Given the set of the top  $Z$  frequency edges after sampling, the *detection accuracy of causal edges* is the number of causal edges in the set divided by  $Z$ , and the *detection accuracy of attack edges* is the number of attack edges in the set divided by  $Z$ .

For each experiment, we use the parameter values selected from Figure 10, and discuss how we compute the optimal parameter values in Section 7.4. We repeat each experiment run 5 times with each run consisting of  $10^4$  walks (unless otherwise specified) and plot the mean of the 5 runs for the following results.

### 7.1 Detecting the Existence of an Attack

To determine whether the random moonwalk technique can detect if an attack is present,  $10^4$  random moonwalks were performed on Trace-10. Figure 11 shows the number of times each edge was sampled, and the outline of the plot indicates the count of the most frequently sampled edge for each second. The dashed lines indicate the actual attack start time, saturation time, and the attack finish time. The figure shows that edges occurring before and after the attack have a relative evenly distributed sampling frequency. Edges between time 2700 and 10000 are sampled more frequently, with a peak frequency as high as 800. This strongly suggests the existence of abnormal structures in the host

Trace	Worm inter-scan duration (second)					
	10	20	30	50	75	100
Normalized worm rate	2	1	0.67	0.4	0.27	0.2
Total flows $ E $ (million)	2.02	1.67	1.57	1.49	1.43	1.42
Number of hosts infected	804	804	804	804	726	702
Fraction of attack edges	0.296	0.157	0.103	0.053	0.013	0.012
Optimal $\Delta t$ (second)	400	800	1600	1600	1600	3200

Figure 10: Description of traces with different rate worm traffic artificially added into a real traffic trace collected from the backbone of a university network.

contact graph, which may potentially constitute an epidemic spreading attack.

In particular, the peak of the frequency counts occurring around 2800 seconds corresponds to the onset of the attack (the worm was introduced at  $T_0 = 2807s$ ) with initial causal flows having highest probability of being traversed. The turning point after the peak (4200 seconds in this case) corresponds to the attack saturation time when all vulnerable hosts are infected. Knowledge that an attack is taking place and the information on precisely when it started is useful to network operators, and could be used to focus resources (such as random moonwalks) on the portions of the trace that are most likely to yield information about the attack origin.

## 7.2 Identifying Causal Edges and Initial Infected Hosts

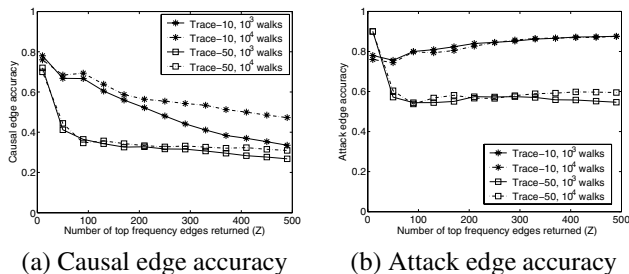


Figure 12: Detection accuracy of causal edges and attack edges vs. number of top frequency edges ( $Z$ ) returned for Trace-10 and Trace-50. Note there are only 800 causal flows from among approximately  $1.5-2 \times 10^6$  total flows.

We first examine the detection accuracy of causal edges and the size of the suspect set identified for further investigation. Figure 12 (a) shows the detection accuracy, varying the number of top frequency  $Z$  edges, with different number of walks. First, we observe random moonwalks achieve high detection accuracy of causal edges, in particular when  $Z$  is small. Although there are only 800 causal edges out of the approximately  $1.5-2 \times 10^6$  flows, as high

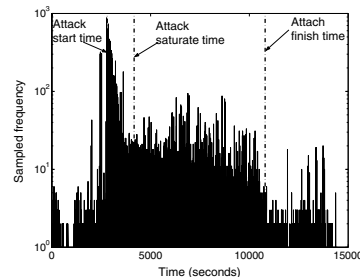


Figure 11: Stem plot of edge frequency counts with  $W = 10^4$  walks on Trace-10.

as 7-8 out of the top 10 flows are causal flows, regardless of the worm propagating rate. Second, the causal edge accuracy decreases sub-linearly as we increase  $Z$ , demonstrating the capability of finding causal flows beyond the few initial ones. These edges may additionally reveal the attack propagation paths, and help reconstruct the causal tree. Finally, increasing the number of walks results in higher causal edge accuracy in general, but a small number of samples can already achieve comparable performance when we focus on the small number of top flows, i.e., when  $Z \leq 100$ . As a contrast, we show the detection accuracy of attack edges in Figure 12 (b). We find that as expected the accuracy of attack edges is fairly high. But a high detection accuracy of attack edges does not always imply high detection accuracy of causal edges. For example, the attack edge accuracy for Trace-10 increases with larger  $Z$ , while the causal edge detection accuracy decreases. In Section 7.5, we will further address the comparison between causal edge and attack edge accuracies with alternative edge selection strategies.

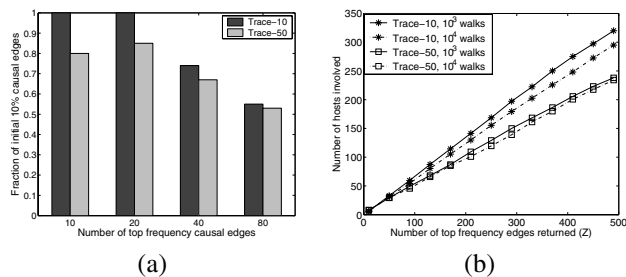


Figure 13: (a) Fraction of initial causal edges among the actual returned causal edges. (b) The number of source hosts involved as suspect top level hosts vs. number of top frequency edges ( $Z$ ) returned.

We proceed to examine whether the detected causal edges correspond to the initial causal edges. We focus on the initial 80 causal flows (10% of the total causal flows) in the attack and plot the fraction of such flows among the actual returned causal edges in Figure 13 (a). As expected, the majority of the causal flows actually detected correspond to

the initial ones that can be traced back to the attack origin, confirming the results in our analytical study.

Given the selected top frequency flows, we examine how many hosts are involved with initiating these flows. Since the identified flows are likely to be top level causal flows, these hosts are good candidates as hosts on the top level causal tree that can be chosen for further investigation. We assume that the source host of every selected flow is potentially the worm origin, and plot the total number of such hosts as we vary the number of selected flows  $Z$  in Figure 13 (b). These numbers thus give an upper bound on the amount of further effort required for worm origin identification (without explicitly exploiting the structure of the graph composed of the selected flows). Although the number of hosts grows linearly as  $Z$  increases, the slope is less than one, suggesting the existence of a small number of sources contributing to a large number of flows. For example, after  $10^4$  walks, if we plan to use the top 50 flows for reconstructing the top level causal tree, we will have in total only 30 source hosts out of the 8040 hosts even with a slowly propagating worm that generates one scan per 50 seconds. In the next section, we show how the structure of the graph composed of these returned high frequency flows can additionally help to identify the worm origin.

### 7.3 Reconstructing the Top Level Causal Tree

Once we obtain the worm origin suspect set and the  $Z$  selected flows, a number of methods could be used to pinpoint the exact attack source. Potential methods include correlating the contents or sizes of the selected flows, or using additional out-of-band information regarding the set of infected hosts. Alternately one can exploit the structure of the graph composed of the  $Z$  flows. We simply take the 60 top-frequency flows selected from Trace-50 after  $10^4$  walks and construct a graph of these flows (Figure 14).

The artificially introduced worm in Trace-50 starts at host 8033, and each infected host sends only one attack flow every 50 seconds. Among the top 60 flows found by random moonwalks and shown in Figure 14, there are 35 causal flows and 17 flows that carry attack traffic but are not the flows that actually caused their destinations to become infected. The random moonwalks identify host 8033 as the actual worm origin and show the large tree branching structure below it. We also observe quite a few flows with destination host 281. It turned out that in the background trace we collected, host 281 was infected by some variant of the Blaster worm [3], and it generates scans with a peak rate of 72 flows per second. Manual investigation into the real trace revealed no successful infection events associated with such scan traffic. As a result, there is no causal tree actually induced by host 281. However, due to the high scanning rate, the few flows sent to host 281 are frequently selected by random moonwalks that trace back to host 281, and this

explains why these normal flows to host 281 appear. Even though there is unrelated aggressive scanning taking place, the random moonwalks still cull out the top levels of the causal tree automatically. Such results show the effectiveness of random moonwalks at extracting the tree structure of slow worm propagation patterns (in our example, one scan every 50 seconds) to identify the worm source, even in the presence of aggressive scanners and other pathological background traffic events. We are currently pursuing refinement techniques to further improve the accuracy of identifying the worm origin(s) and to reconstruct the higher levels of the causal tree.

### 7.4 Parameter Selection

Given a network trace that may contain worm traffic, we need to select the best parameter values without prior knowledge of worm propagating characteristics. This section studies the performance impact of the input parameters  $d$  (maximum path length) and  $\Delta t$  (sampling window size). We use Trace-20 and Trace-50 as representative traces for the following study.

We first fix  $\Delta t$  to 800 seconds for both traces (800 seconds may not be the optimal value for each trace) and vary the maximum path length  $d$  in terms of hop counts. Figure 15 (a) shows the detection accuracy of the top 100 frequency edges (i.e.,  $Z = 100$ ). We observe that the detection accuracy for both attack edges and causal edges increases with longer path length. As discussed earlier in our analysis in Section 6.4, longer paths tend to walk across a larger portion of the attack tree. As we further increase the path length, the detection accuracy saturates as the path length of each walk is bounded by the start of the trace. A longer maximum path length improves detection accuracy, but also implies greater sampling overhead since more edges will be involved in each walk.

Next, we vary the sampling window size  $\Delta t$  with the maximum path length  $d$  set equal to  $\infty$  so each walk can continue as far as possible. Figure 15 (b) shows the impact of  $\Delta t$  on the detection accuracy of the 100 top frequency edges. In both traces, when we increase  $\Delta t$ , the detection accuracy of the causal edges first increases and then decreases. The detection accuracy of attack edges, however, is highest for smaller  $\Delta t$ 's and becomes lower with a larger  $\Delta t$ . We also observe that with the slowly propagating worm in Trace-50, we need a larger  $\Delta t$  to achieve the best detection accuracy compared with the faster propagating worm in Trace-20.

To understand the reason, we show in Figure 15 (c) the variation of the actual path lengths (in terms of hop-count) with  $\Delta t$ . When  $\Delta t$  is small, walks terminate at shorter path lengths, as a walk is more likely to reach a host that received no flows within the previous  $\Delta t$  seconds. While shorter walks cannot reach the top levels of the causal tree, they

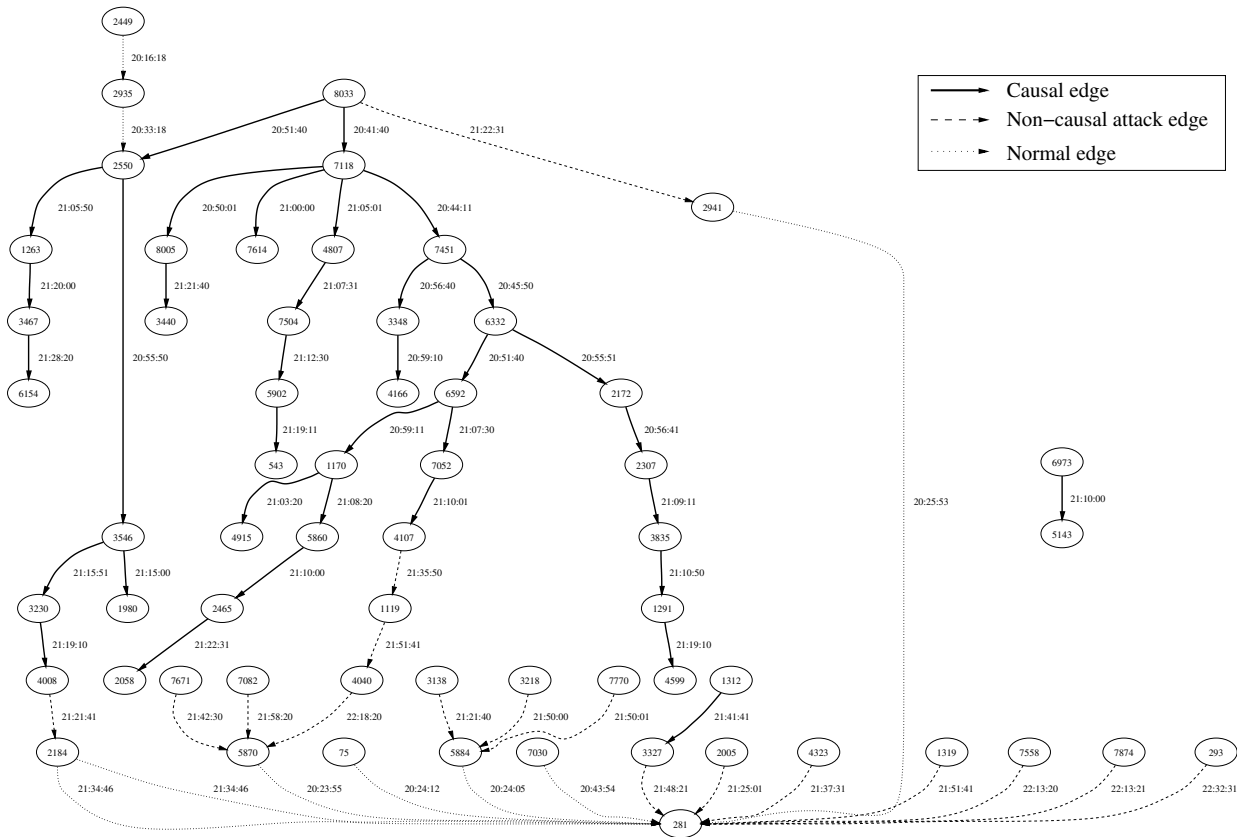


Figure 14: Graph of the 60 top frequency flows returned by the random moonwalk algorithm when run on Trace-50. Note the graph is neither the host contact graph, nor the causal tree. Hosts are represented by circles annotated with the host ID. Flows are represented as directed arrows between hosts, and are annotated with timestamps. Solid arrows denote causal edges, dashed arrows denote non-causal attack edges, and dotted edges correspond to normal traffic flows.

are more likely to stumble across attack edges at lower levels, resulting in high detection accuracy for attack edges but low accuracy for causal edges. Increasing  $\Delta t$  gives a random moonwalk a greater chance to traverse top level edges, in particular the causal ones, but these long paths also involve more normal flows since they can walk backward to before the start of the attack, reducing the number of attack edges involved. Thus the detection accuracy of causal edges increases while that of attack edges decreases. Finally, further increasing  $\Delta t$  has a negative impact on the actual lengths of walks as each walk tend to be shorter by jumping across a larger portion of the trace every step. The walks also involve more normal traffic, since attack flows are generally clustered in time and a large  $\Delta t$  can skip over large portions of the attack. As a result, we observe low detection accuracy for both types of edges when  $\Delta t$  is too large.

For both Trace-20 and Trace-50, we achieve the best detection accuracy for causal edges when actual path lengths are maximally long. For worms that generate flows with a

slower rate, a larger  $\Delta t$  maximizes the actual path lengths and achieves better performance.

In summary, given a trace with unknown worm properties, the best sampling performance is obtained by choosing the  $\Delta t$  that gives the longest actual path lengths, in terms of number of hops that the moonwalks traverse. For all our experiments, we used the above guideline to choose an optimal  $\Delta t$  for each trace (see Figure 10). An adaptive version of random moonwalk sampling could launch walks with different values of  $\Delta t$  and choose one that maximizes the observed path lengths.

## 7.5 Performance vs. Worm Scanning Rate

In this experiment we compare the random moonwalk algorithm with other common methods for identifying potentially anomalous behavior, while varying the rate at which infected hosts scan new victims. Again, we use the detection accuracy of both causal and attack edges as our performance metrics, and we compare the following five techniques:

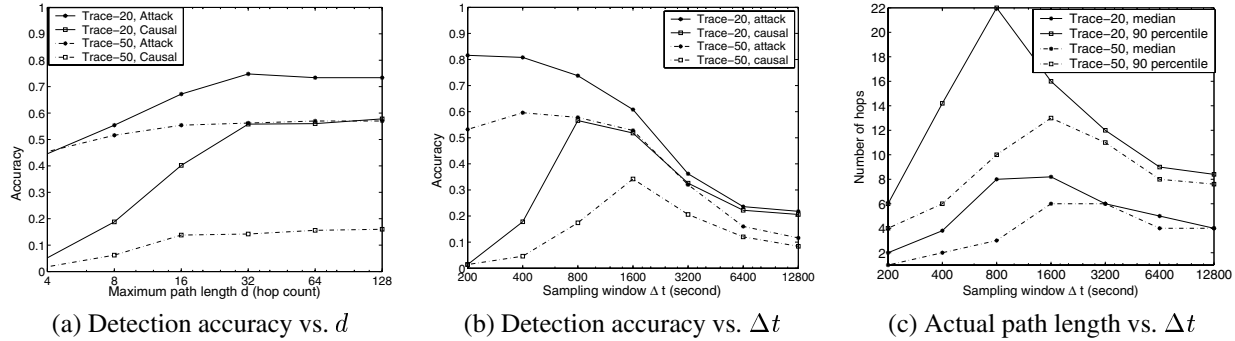


Figure 15: Impact of parameter selection on performance using both Trace-20 and Trace-50.

- **Random moonwalk selection:** Pick the  $Z = 100$  edges with the highest frequency after performing  $10^4$  random moonwalks.
- **Heavy-hitter detection:** Find the 800 hosts that generated the largest number of flows in the trace (the “heavy-hitters”). Randomly pick 100 flows between two heavy-hitters. (We select 800 hosts as we know there are about 800 infected hosts in the traces.)
- **Super-spreader detection:** Find the 800 hosts that contacted the largest number of distinct destination hosts (the “super-spreaders”). Randomly pick 100 flows between two super-spreaders.
- **Oracle selection:** Assume an oracle that identifies the set of infected hosts with zero false positive rate. The oracle randomly selects 100 flows between these hosts.
- **Random selection:** Randomly pick 100 flows from each trace.

Both heavy-hitter and super-spreader heuristics have been traditionally used to detect patterns of malicious activity in IDSes [17, 18].

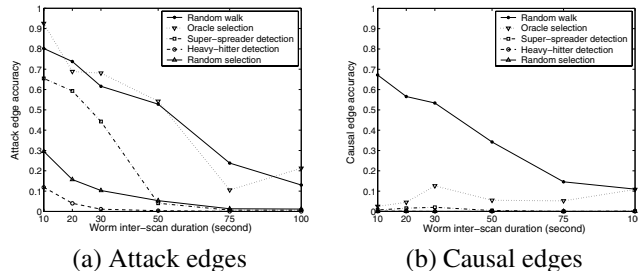


Figure 16: Detection accuracy vs. worm scanning rate. The X-axis represents the worm inter-scan duration. For example, a window of  $x = 20$  means an infected host generates an infection flow every 20 seconds.

As expected, the detection accuracy for attack edges decreases with an increased worm inter-scan duration (Figure 16 (a)), since a worm that sends attack traffic at a slower rate will create fewer attack edges in the host contact graph. Random moonwalk selection and oracle selection have similar performance and perform substantially better than the other strategies. Perhaps surprisingly, heavy-hitter detection performs even worse than random selection, as the heavy-hitter method is likely to select servers, and most of the communication between servers is legitimate traffic.

The real success of the random moonwalk algorithm, however, is not in picking attack edges. Rather it lies in its ability to extract causal edges from a large noisy host contact graph. This is evident from Figure 16 (b), where we notice that all other techniques, including oracle selection, have a low detection accuracy for causal edges across all worm scanning rates. For attacks that spread at rates of one scan every 10-30 seconds, the causal edge detection accuracy of random moonwalk selection is greater than 0.5, implying that roughly 50 out of the top 100 edges are always causal edges. This establishes the capability of finding the causal edges by globally correlating the host traffic patterns for very stealthy attacks using the random moonwalk algorithm. On the other hand, the poor performance of even the oracle selection suggests that detecting infected hosts alone does not help extracting the causal edges to reconstruct the top level causal tree and trace back the worm origin.

## 7.6 Performance vs. Worm Scanning Method

In this experiment, we study the effectiveness of random moonwalks using worms with different scanning methods. Since many existing techniques identify worm scanners by looking at only flows sent to non-existent hosts [11, 28], a smart worm can evade such detection by carefully targeting only valid addresses. We therefore evaluate the performance of our technique using two worms with different scanning methods. The first scanning method randomly scans only valid host addresses, while the second method randomly scans both existent and non-existent host

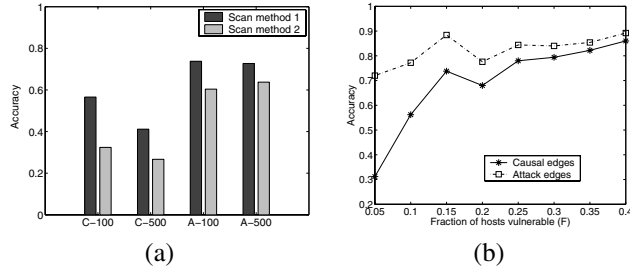


Figure 17: (a) Comparing detection accuracy using worms with different scanning methods using Trace-20. (b) Comparing detection accuracy using worms targeting different fraction of vulnerable hosts  $F$ .

addresses with 50% of host address space being used. For both worms, an infected host starts scanning at the rate of one attack flow every 20 seconds.

Figure 17 (a) compares the detection accuracy of the top  $Z = 100$  and  $Z = 500$  frequency edges for the two different worms. For both causal edges (represented by C-100 and C-500) and the attack edges (represented by A-100 and A-500), random moonwalks achieve better detection accuracy for the “smart-scanning” worm, which is consistent with our analytical study in Section 6.2. As random moonwalk sampling identifies the subtle global tree patterns of worm propagation, instead of relying on the scanning behavior of each specific infected host, it is inherently more robust to other worm scanning strategies [25, 28]. Such results are also encouraging for detecting those worms that may evade detection techniques employed by many existing scan-detectors, which essentially use the number of connections to unused address chunks as a metric of interest [11, 12, 20].

### 7.7 Performance vs. Fraction of Hosts Vulnerable

This section studies the performance of the random moonwalk algorithm with different fraction of hosts infected (i.e., we vary  $F$ ). With a greater number of hosts infected by an attack, the degree of anonymity provided to the true attacker is also greater. In this experiment, we fix the worm scanning rate to be one attack flow per 20 seconds, and vary the fraction of hosts vulnerable  $F$  during each attack. Figure 17 (b) shows the performance in terms of the detection accuracies of both causal edges and attack edges. Within the range of  $F = [0.05, 0.4]$ , we observe that the detection accuracies increase as we increase the fraction of hosts infected. Empirically, our experiments also show that the detection accuracy increases for more slowly propagating attacks (e.g., one scan per 50 seconds) as they infect more hosts in the network along time. We plan to further quantify the impact of  $F$  on performance as future work.

## 8 Simulation Study

The goal of our simulation study is to evaluate the effectiveness of random moonwalks using different background traffic models of normal host communication. Our hypothesis is that the simplified traffic model in our analytical study, where background (i.e., normal) traffic, modeled as uniform scanning, is a worst case model for performance of our algorithm. Realistic host contact graphs tend to be much sparser, meaning the chance of communication between two arbitrary hosts is very low since host connectivity patterns usually display locality in the set of destinations contacted. An epidemic “tree” structure will more easily stand out in such scenarios, and thus be detected with higher accuracy.

In particular, we model the host connectivity patterns in terms of both the out-degree of normal hosts and the connection locality. The out-degree of each normal host is the size (denoted as  $D$ ) of the *contact set*, which represents the set of destinations the host originates flows to under normal circumstances. Connection locality is modeled by assuming each host selects destinations preferentially (within the *contact set*) according to either a uniform or power-law distribution. Figure 18 lists the background traffic generated using different combinations of the host out-degree and connection locality. All the simulations run with  $|H| = 10^4$  nodes for 3000 seconds of simulated time. We introduce worm attacks lasting 500 seconds with a fixed propagating rate ( $A/B \simeq 7$ ) that infect  $F = 0.1$  fraction of hosts. Recall that  $A$  is the connection rate of an infected host (including normal connections), and  $B$  is the connection rate of a normal host. The resulting traces have about  $10^6$  total flows with 1000 causal flows. For each trace, we perform  $10^4$  random moonwalks and compute the detection accuracy of causal edges among the returned top  $Z = 100$  frequency flows.

Overall, the random moonwalks achieve high detection accuracy across all background traffic models. As expected, the power-law distribution of the host out-degree results in best performance as the corresponding normal host contact graphs are sparse. The power-law distribution connection locality has similar performance impact since each host tends to talk only to a few hosts within the contact set more frequently, resulting in a relatively sparser host contact graph too. In contrast, uniform destination selection with constant contact set size (i.e.,  $D = C$ , or  $D = |H|$ ) models random scanning background traffic, and yields the worst performance.

## 9 Deployment and Future Work

Similar to single-packet IP traceback [23], we envision an architecture in which distributed collection points log flow records and corresponding timestamps, and store them in repositories for querying. In addition to the source and des-

Trace	1	2	3	4	5
Host out-degree $D$	$ H $	$C <  H $	$ H $	Power-law	Power-law
Connection locality	Uniform	Uniform	Power-law	Uniform	Power-law
Causal edge accuracy	0.506	0.472	0.546	0.616	0.614

Figure 18: Detection accuracy of causal edges using different background traffic models. “Power-law” means the controlled variable follows a power-law distribution (with a coefficient set to 3). “Uniform” means the controlled variable follows a uniform distribution.  $|H|$  denotes the total number of hosts in the network, and  $C$  is a constant number smaller than  $|H|$ .

mination IP addresses, each flow record contains an identifier for distinguishing two flows between the same source and destination at roughly the same time, for which we can use, e.g., the 13-bit identifier field of the initial packet in the flow in the case of IPv4. Though this is not strictly necessary, it permits us to relax the degree of clock synchronization necessary among collection points and can improve the accuracy of our search. At each individual collection point, we require two causally related flows be logged in their causal order and timestamped with a common clock.

As in single-packet IP traceback [23], a concern for traffic logging is whether the storage capacity required will be excessive. A back-of-the-envelope calculation in [21] suggests that the amount of flow level storage requirement is not inconceivable, even for a large Tier-1 ISP. We also note that by the time a worm infection becomes so pervasive, that the induced traffic potentially outpaces these logging capabilities, the records most important for finding the attack origin, namely those close to the origin, have already been recorded.

Our approach is effective for the class of attacks that propagate via “tree” structured communication patterns. Future work includes the development of algorithms to perform post-mortem analysis of a larger class of attacks. Our current implementation assumes that the semantic direction of the flow is consistent with the network notion of flow directionality. Attacks may try to obfuscate the notion of causality among network flows. We are currently exploring ways to make the algorithm robust to such attacks. Our approach currently assumes the availability of complete data. It is likely that traffic auditing will be deployed incrementally across different networks. We are investigating the impact of missing data on performance, and also the potential for incremental deployment of the algorithm. Our initial results in this direction have been promising.

## 10 Conclusions

In this paper, we present the random moonwalk algorithm to identify the origin or the entry point of epidemic spreading attacks by identifying the initial successful infection flows. Our approach explores the globally visible tree-like structure of worm propagation using flow-level records logged by the networks. By ignoring packet-level characteristics

and attack signatures, our algorithm is potentially agnostic to attack specific characteristics such as payload contents, port numbers used, or specific software vulnerabilities exploited. Our analysis, simulation based experiments, and real trace study demonstrate that the algorithm is effective in identifying the causal relationships between initial infection events to reveal the worm origin with low false positive rates. We also demonstrated that the algorithm is robust to low-rate attacks trying to masquerade as normal traffic, or smart scanning worms that may evade known scan-detection techniques.

## References

- [1] P. Barford, J. Kline, D. Plonka, and A. Ron. A Signal Analysis of Network Traffic Anomalies. In *Proc. of ACM SIGCOMM Internet Measurement Workshop*, 2002.
- [2] S. Bellovin, M. Leech, and T. Taylor. ICMP Traceback Messages. Internet draft, work in progress, 2001.
- [3] CERT Advisory CA-2003-20: W32/Blaster worm. <http://www.cert.org/advisories/CA-2003-20.html>, 2003.
- [4] A. Blum, D. Song, and S. Venkataraman. Detection of Interactive Stepping Stones: Algorithms and Confidence Bounds. In *Proc. of The Seventh International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2004.
- [5] J. Browne. Probabilistic Design. <http://www.ses.swin.edu.au/homes/browne/probabilisticdesign>.
- [6] H. Burch and B. Cheswick. Tracing Anonymous Packets to Their Approximate Source. In *Proc. of USENIX LISA Systems Administration Conference*, 2000.
- [7] D. L. Donoho, A. G. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford-Chen. Multiscale Stepping-Stone Detection: Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay. In *Proc. of The 5th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2002.
- [8] The Dragnet Project. <http://www.cs.cmu.edu/~dragnet>.
- [9] P. Ferguson and D. Senie. RFC 2267 - Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing, 1998.
- [10] A. Hussain, J. Heidemann, and C. Papadopoulos. A Framework for Classifying Denial of Service Attacks. In *Proc. of ACM SIGCOMM*, 2003.

- [11] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In *Proc. of IEEE Symposium on Security and Privacy*, 2004.
- [12] H. A. Kim and B. Karp. Autograph: Toward Automated, Distributed Worm Signature Detection. In *Proc. of 12th USENIX Security Symposium*, 2004.
- [13] C. Kreibich and J. Crowcroft. Honeycomb – Creating Intrusion Detection Signatures Using Honey Pots. In *Proc. of ACM HotNets-II*, 2003.
- [14] J. Kubica, A. Moore, D. Cohn, and J. Schneider. Finding Underlying Connections: A Fast Graph-Based Method for Link Analysis and Collaboration Queries. In *Proc. of Twentieth International Conference on Machine Learning*, 2003.
- [15] J. Li, M. Sung, J. Xu, L. Li, and Q. Zhao. Large-scale IP Traceback in High-speed Internet: Practical Techniques and Theoretical Foundation. In *Proc. of IEEE Symposium of Security and Privacy*, 2004.
- [16] D. Moore, G. M. Voelker, and S. Savage. Inferring Internet Denial-of-Service activity. In *Proc. of 10th USENIX Security Symposium*, 2001.
- [17] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. In *Proc. of 7th USENIX Security Symposium*, 1998.
- [18] M. Roesch. Snort - Lightweight Intrusion Detection for Networks. In *Proc. of USENIX LISA Systems Administration Conference*, 1999.
- [19] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical Network Support for IP Traceback. In *Proc. of ACM SIGCOMM*, 2000.
- [20] S. E. Schechter, J. Jung, and A. W. Berger. Fast Detection of Scanning Worm Infections. In *Proc. of 7th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2004.
- [21] V. Sekar, Y. Xie, D. Maltz, M. K. Reiter, and H. Zhang. Toward a Framework For Internet Forensic Analysis. In *Proc. of ACM HotNets-III*, 2004.
- [22] A. Snoeren. Public review of ‘Toward a Framework for Internet Forensic Analysis’. In *Proc. ACM HotNets-III*, 2004.
- [23] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. Hash-Based IP Traceback. In *Proc. of ACM SIGCOMM*, 2001.
- [24] S. Staniford-Chen and L. T. Heberlein. Holding Intruders Accountable on the Internet. In *Proc. of the IEEE Symposium on Security and Privacy*, 1995.
- [25] S. Staniford-Chen, V. Paxson, and N. Weaver. How to Own the Internet in Your Spare Time. In *Proc. of 11th USENIX Security Symposium*, 2002.
- [26] X. Wang and D. Reeves. Robust Correlation of Encrypted Attack Traffic Through Stepping Stones by Manipulation of Inter-packet Delays. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, 2003.
- [27] R. J. Wonnacott and T. H. Wonnacott. Introductory Statistics. *Fourth Edition*.
- [28] J. Wu, S. Vangala, L. Gao, and K. Kwiat. An Effective Architecture and Algorithm for Detecting Worms with Various Scan Techniques. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2004.
- [29] Y. Zhang and V. Paxson. Detecting Stepping Stones. In *Proc. of 9th USENIX Security Symposium*, 2001.

## Appendix: Probability Estimation in Section 6.2

An edge  $e = \langle u, v, k \rangle$  can occur at different steps of a random moonwalk. We use  $P_i(e^k)$  to denote the probability of an edge at time  $k$  being traversed by the  $i$ -th step of a walk. Then we have  $P(e^k) = \sum_{i=1}^d P_i(e^k)$ .

We use  $O(v, k)$  to denote the number of concurrent outgoing flows from host  $v$  at time  $k$ . With  $|E|$  edges in the host contact graph, we have

$$P_i(e^k) = \begin{cases} 1/|E| & i = 1 \\ \left( \sum_{j=1}^{O(v, k+1)} P_{i-1}(e^{k+1, j}) \right) / I(v, k) & i > 1 \end{cases}$$

where  $e^{k+1, j}$  is the  $j$ th flow generated by host  $v$  at time  $k+1$ , and  $I(v, k)$  is the number of incoming flows into host  $v$  at time  $k$ . The above equation holds for any host contact graph, without any assumptions. Under the uniform scanning assumption for both normal and attack traffic, a second order approximation for  $E\left(\frac{1}{I(v, k)}\right)$  is ,

$$\begin{aligned} E\left(\frac{1}{I(v, k)}\right) &= \frac{1}{E(I(v, k))} \left( 1 + \left[ \frac{\sigma_{I(v, k)}}{E(I(v, k))} \right]^2 \right), \text{ from [5].} \\ &\approx \frac{1}{E(I(v, k))} = \frac{1}{I(k)} \end{aligned}$$

The above approximation holds for large enough  $|H|$  and  $A$ , since  $I(v, k)$  is binomially distributed.

Under the simplified assumptions discussed in Section 6.1, if  $e_m^k = \langle u, v, k \rangle$  is a malicious-destination edge, we have  $O(v, k+1) = A$ , otherwise,  $O(v, k+1) = B$ . Using the approximate form for  $1/I(v, k)$  above, for an edge at time  $k$  we have:

$$\begin{aligned} P_2(e_m^k) &\approx \frac{1}{I(k)} \sum_{j=1}^A P_1(e^{k+1, j}) = \frac{A}{|E|I(k)} \\ P_2(e_n^k) &\approx \frac{1}{I(k)} \sum_{j=1}^B P_1(e^{k+1, j}) = \frac{B}{|E|I(k)} \\ P_3(e_m^k) &\approx \frac{1}{I(k)} \sum_{j=1}^A P_2(e^{k+1, j}) = \frac{(B+R)T_{k+1}}{|E|I(k)I(k+1)} \\ P_3(e_n^k) &\approx \frac{1}{I(k)} \sum_{j=1}^B P_2(e^{k+1, j}) = \frac{B \times T_{k+1}}{|E|I(k)I(k+1)} \end{aligned}$$

By induction, we can easily show that  $\forall d' (4 \leq d' \leq d)$ ,

$$\begin{aligned} P_{d'}(e_m^k) &\approx \frac{(B+R)T_{k+d'-2}}{|E|I(k)I(k+1)} \\ P_{d'}(e_n^k) &\approx \frac{BT_{k+d'-2}}{|E|I(k)I(k+1)} \end{aligned}$$

Taking the sum of all  $P_i(e)$  ( $1 \leq i \leq d$ ), we have

$$\begin{aligned} P(e_m^k) &\approx \frac{1}{|E|} \left[ 1 + \frac{A}{I(k)} + \frac{(B+R) \times \sum_{i=1}^{d-2} T_{k+i}}{I(k)I(k+1)} \right] \\ P(e_n^k) &\approx \frac{1}{|E|} \left[ 1 + \frac{B}{I(k)} + \frac{B \times \sum_{i=1}^{d-2} T_{k+i}}{I(k)I(k+1)} \right] \end{aligned}$$