On the Detection and Origin Identification of Mobile Worms

Sandeep Sarat Andreas Terzis
Computer Science Department
Johns Hopkins University
Baltimore, MD, USA
{sarat,terzis}@cs.jhu.edu

ABSTRACT

Mobility can be exploited to spread malware among wireless nodes moving across network domains. Because such *mobile worms* spread across domains by exploiting the physical movement of mobile nodes, they cannot be contained by existing defenses. In this paper we address this new challenge using techniques for detecting the existence of stealthy mobile worms in the early stages of their infection and identifying the origins of such infections.

The proposed mechanisms are based on random moonwalks which were originally used in post mortem analysis of Internet worms. However as we demonstrate, the original technique fails against mobile worms which are inherently stealthier than existing malware. In this paper, we extend the moonwalk algorithm by considering new heuristics and show that the proposed mechanism can reliably detect mobile worms during the early stages of infection. Our simulation results, based on network traces collected from a university-wide wireless network, show that a mobile infection can be reliably detected before it infects 10% of the vulnerable population. Furthermore, the proposed mechanism identifies the origin of the infection, by limiting the search for the initial victims to within 2% of the mobile node population.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Invasive Software(e.g., viruses, worms, Trojan horses)*

General Terms

Security, Measurement

Keywords

Network Security, Internet worms, Methods of Attribution, Mobility

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
WORM'07, November 2, 2007, Alexandria, Virginia, USA.
Copyright 2007 ACM 978-1-59593-886-2/07/0011 ...$5.00.

1. INTRODUCTION

Mobility pervades networked devices today. An increasing number of users access the Internet through WiFi-equipped devices connected to Access Points (APs) located on campuses, coffee shops, airports, etc. Unfortunately, this increased mobility provides another propagation vector for spreading malware. As a matter of fact, the threat against mobile nodes is not just our speculation. For example, the Zotob worm [19] scans port 445 to infect its victims. Since most enterprises block port 445 at their perimeter, it is likely that the infection propagated via laptops which were infected while outside the corporate network and subsequently infected other machines after connecting to the intranet. While there is no proof that this effect was intended by Zotob's authors, we believe that it is trivial to engineer a worm which exploits physical mobility to propagate. Furthermore, it was shown that such mobile contagions can attain epidemic proportions within a day [2,12], thus rendering manual defenses implausible.

Since mobile worms cross network boundaries via infected nodes that physically migrate across domains, it is difficult to contain them when no controls exist to police the movement of nodes across domains. Furthermore, the detection of these worms is challenging because the majority of techniques against zero-day infections rely on recognizing anomalous patterns in inbound traffic (*e.g.*, [3] among others), or outbound DNS, ARP, or failed connection requests from local hosts [13,16]. On the other hand, a mobile worm finds victims by eavesdropping on the radio channel, thus generating no scanning/ARP/DNS traffic. Moreover, the alternative solution of using honeypots for detection is also ineffective because it requires placing honeypots in the majority of the domains [12].

In this paper, we present two mechanisms for countering mobile worms. The first mechanism detects the existence of a worm spreading through a collection of wireless domains while the second identifies the origin of the worm. Doing so involves identifying the node(s) that initiated the infection as well as the nodes infected during the very early stages of the epidemic. In turn, origin identification enables further investigation into the underlying causes and techniques used to breach the networks defenses and can provide information relevant to law enforcement. These node identities can also be used to contain the infection, by blocking their traffic or by automatically generating attack signatures based on the traffic they transmit.

Both proposed mechanisms extend the Random Moonwalk technique [17] and only require network flow records

consisting of the start, duration, source, and destination of all flows within a wireless network domain. These flow records are collected at every domain and are either aggregated to a centralized database, or are available through a federated database, similar to the network forensic alliance (NFA) proposed in [18]. We first show that the original moonwalk is ineffective against mobile infections and then present two new heuristics that can detect and identify such infections. We evaluate the performance of the proposed algorithms through simulations driven from network traces collected from a university-wide wireless network. Our results show that a mobile infection can be reliably detected before it infects 10% of the vulnerable population in a network with hundreds of domains and thousands of mobile nodes. Furthermore, the proposed identification algorithm limits the search for the initial infection victims to within 2% of the mobile node population. Working in concert, the two algorithms we present can effectively protect users against stealthy mobile worms.

This paper has six sections. In the following section, we present the network model we use and review the standard moonwalk algorithm. Section 3 describes how the moonwalk algorithm can be modified for the online detection of mobile worms. In Section 4 we show how to trace the evolution path of a mobile worm. Finally, Section 5 presents related work and we close in Section 6.

2. BACKGROUND

2.1 Network and Worm Models

The network model we use consists of two types of entities: **(a)** network domains that mobile nodes connect to (*e.g.* (APs) in the case of WiFi networks), and **(b)** mobile nodes that are susceptible to infections and move across these domains. This model encompasses real-life scenarios such as metropolitan mesh networks serving mobile users and enterprise networks that host employees' or visitors' laptops etc. These networks are susceptible to self replicating infections which use mobility as a propagation vector. Infected nodes find victims by eavesdropping on other nodes' transmissions. Moreover, infected machines only attempt to subvert nodes in the domain they currently reside. The novelty of this type of attack is that the infection spreads across network domains through the physical movement of infected nodes. Akritidis *et al.* use the term *wildfire worms* for these attacks [1].

As part of our previous work we showed that such mobile infections can spread through tens of thousands of victims located in hundreds of domains within a day [12]. We also showed that a mobile infection initially "moves" towards highly popular domains, because many nodes visit them. When the infection reaches these popular domains, its growth rate rapidly increases and in the final phase it slowly spreads to the remaining domains.

Intuitively, placing network monitors and honeypots in the most popular domains yields the earliest detection times. In fact, we showed that by installing monitors in $\sim$10% of the most popular domains, one could detect the infection while it is still in its early phase. Deploying such wireless network monitors involves modifying APs to inspect the packets they forward or deploying honeypots acting as mobile nodes. Unfortunately, deploying monitors in the most popular domains is insufficient for a number of reasons. First, domain popularity changes over time depending on the users' mobility patterns. Second, mobile worms can potentially detect the presence of monitors and avoid popular domains in which they may be deployed. Similar *probe-response attacks*, used to discover the locations of network monitors deployed on the (wired) Internet have been discussed in [4, 11]. Mobile worms could use standard tools such as sending `ICMP` or `ARP` requests, or even eavesdrop to infer the size of a domain and avoid highly popular domains. Finally, worm origin identification is almost impossible if monitors are not deployed in every domain.

2.2 Random Moonwalks

The random moonwalk is a post-mortem method for identifying the origins of a worm attack on the Internet using network flow data [17]. Specifically, given a set of network flow records corresponding to a host contact graph, a moonwalk starts at an arbitrarily chosen edge $e_1 = \langle u_1, v_1, t_1^s, t_1^e \rangle$, where u_1, v_1 are the source and destination and t_1^s, t_1^e, the start and end times of the flow respectively. The next edge backward in time is selected uniformly at random from the set of edges that arrived at u_1 within the past Δt seconds, *i.e.* $e_2 = \langle u_2, u_1, t_2^s, t_2^e \rangle$ and $t_2^e < t_1^s < t_2^e + \Delta t$. This process continues for a maximum number of hops or until no prior edge is found. Multiple moonwalks are taken and the edges that appear with the highest frequency across all moonwalks are computed. These edges are likely to be the top-level causal edges of the worm tree (*i.e.* edges that initiated the infection). The intuition behind this approach is that because worms generate tree-like contact graphs in which a small number of early malicious edges tend to be responsible for a large number of edges further down the tree, the initial causal edges will be traversed multiple times and will therefore have high occurrence frequencies.

The effectiveness of the moonwalk algorithm decreases as the worm becomes stealthier, thus generating smaller amounts of excess traffic. For this reason tracking mobile worms is especially challenging for the moonwalk algorithm, as infected nodes eavesdrop to discover victims instead of scanning for them. Because an infected node only seeks victims within its current domain, it is enough for the moonwalk algorithm to focus on the intra-domain flows. However, even considering this reduced edge set, edge frequency is not a reliable indicator of the initial causal edges of a mobile infection. In fact, as we shall show shortly, it is impossible to even detect the presence of a mobile infection using the standard moonwalk algorithm. The underlying reason is that a typical host contact graph is globally sparse but has considerable local correlation. In other words, the density of flows between nodes in then same domain is higher than those across domains, for both the worm and non-malicious traffic. Moreover, global and local contacts are made at different timescales. While local contacts occur in the timescales of normal host connections, the timescales of inter-domain contacts are governed by the, usually slower, physical movement of nodes across domains.

3. MOBILE WORM DETECTION

3.1 Random Moonwalks and Mobile Worms

To demonstrate the shortcomings of the standard moonwalk algorithm, we simulate a mobile infection that spreads over a group of mobile nodes traversing multiple network

Description	Setting
Number of domains	626
Number of mobile nodes	6101
Flow Inter-arrival Model	Lognormal
Flow Duration Model	Bi-Pareto
Mean Domain Residence Time (T_R)	67 min
Mean OFF Time	315 min
Moonwalk Window Size (Δt)	300 min
Maximum Moonwalk Hop Count (d)	50

Table 1: Simulation Parameters.

domains. We do so using two models: one describing the mobility pattern of nodes across domains and another reflecting the traffic patterns of mobile nodes within a domain. We derive the first model from traffic traces collected at the university campus in Dartmouth, available through CRAWDAD [8]. The trace we use contains 626 domains (*i.e.*, APs) and over 6,000 nodes and was collected from a campus-wide WiFi network between 9/23/2003 and 12/10/2003. Each trace entry corresponds to the time that a host, identified by its MAC address, connected to one of the network domains. The trace also includes the special 'OFF' location, signifying a host's departure from the network.

To the best of our knowledge, there are no datasets that capture traffic that originates and terminates within the same wireless domain. For this reason we generate traffic synthetically. Specifically, we build a flow model using measurements of intra-domain traffic, collected over a period of two weeks from the wireless APs in the Information Security Institute at Johns Hopkins University. Note that applications such as FTP create two TCP flows, one for control messages and one for data. We combine all the flows corresponding to the same transaction into a single *semantic* flow. We then model semantic flow inter-arrival times using a Lognormal distribution and flow sizes using a bi-Pareto distribution, as suggested by [5]. The parameters for these distributions are fitted from the collected packet traces. Finally, we select the size of the moonwalk's time window Δt to maximize the walk lengths and set the hop count value to a large value, to allow the moonwalks to continue as far back as possible. Table 1 summarizes all the parameters used in our simulations.

Given these parameters, we simulate two scenarios: one with only normal traffic and one in which a mobile worm is injected at a random network node at a certain point in time. We let both simulations progress until the time when the worm has infected 65% of the network's nodes and then invoke the random moonwalk algorithm for each of the two scenarios[1].

The top panel in Figure 1 presents the results of the random moonwalk algorithm for the network with no malicious traffic, while the bottom half presents the results when a mobile infection is injected at $t = 10,000$ sec ($\sim$ 167 min). The y-axis in these graphs corresponds to the frequency with which a flow that starts at certain point in time occurs over the set of random moonwalks performed. Therefore, a large frequency value indicates an edge that was traversed during multiple moonwalks.

When the moonwalk algorithm executes on an Internet

[1]Similar results were derived for other infection percentages.

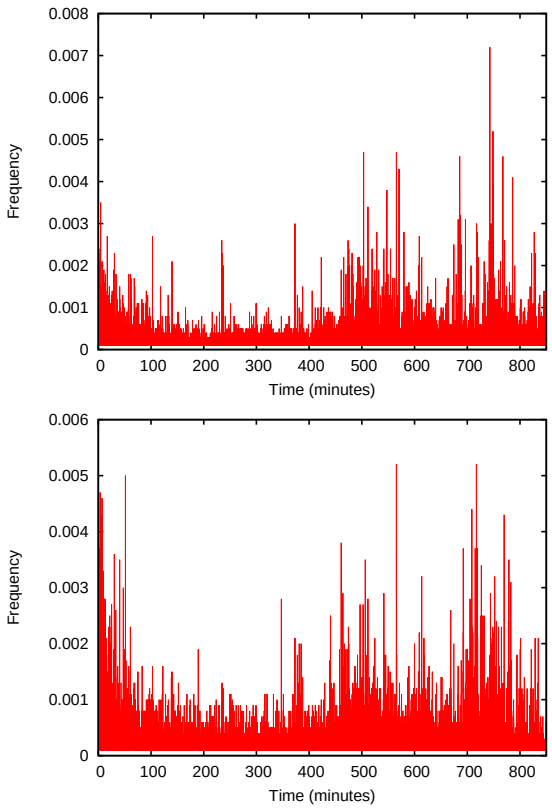

Figure 1: (a) Random moonwalk on a network with no malicious traffic. (b) Random moonwalk on the same network when a worm is injected at $t \sim 167$ min. The y-axis represents the frequency with which flows starting at a particular time, appear in the set of paths traversed by the moonwalks.

trace that contains an actively spreading scanning worm, the initial causal edges of the attack have the highest frequencies creating a pronounced spike in the frequency graph (see [17]). In contrast, as is evident from Figure 1, there is no marked increase in edge frequencies when a mobile worm is spreading. Comparing the two cases, it is difficult to even infer the existence of a worm in the lower graph.

3.2 Proposed approach

As the results from the previous section indicate, edge frequency is not an effective indicator of infection in mobile networks. Instead, we use a different heuristic—the average moonwalk length. The intuition for selecting this attribute is that as the worm spreads, the host contact graph becomes inherently more dense as infected nodes contact other nodes to spread the infection. As a result, the length of the moonwalk tends to increase. Moreover, these contact paths tend to span multiple network domains, which is unusual for normal traffic. Based on these observations, we posit that a worm can be detected by noticing a steep increase in the average moonwalk length.

To test the validity of this thesis, we compute the average moonwalk length for the simulated network presented in the previous section and observe whether the introduction of a worm creates a marked increase in the average moonwalk

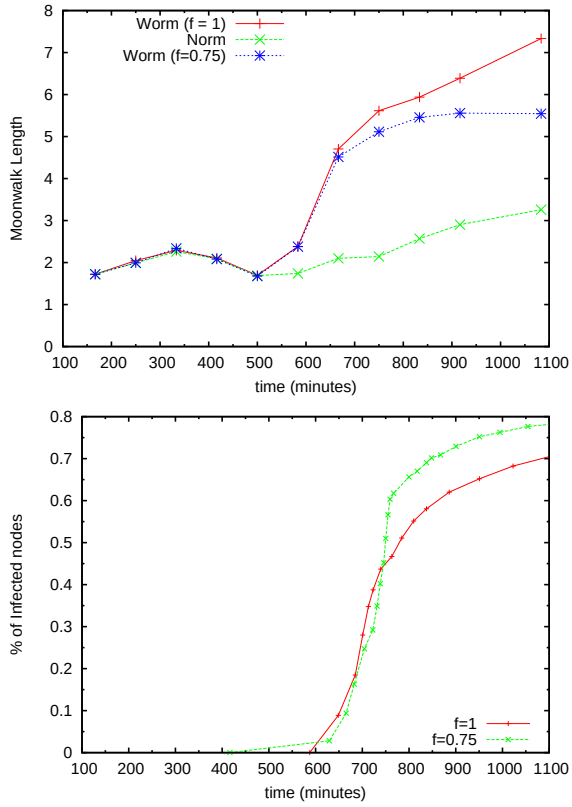

Figure 2: (a) Average moonwalk length for a network with no malicious traffic and a network in which a worm is injected at $t \sim 420$ min. Graphs are shown when 100% and 75% of the population is vulnerable (b) Percentage of infected nodes as a function of time for the same worm.

length. To make the worm even more stealthy, we even allow infected hosts to avoid contacting nodes which they had either previously subverted or attempted to subvert.

Figure 2.(a) presents the average moonwalk lengths for a network with no malicious traffic and a network in which a worm is injected at time $t = 25,000$ sec ($\sim$ 420 min). It is clear from this graph that the average moonwalk length increases considerably after the infection starts. At approximately 650 minutes into the simulation, the walk length in the worm scenario is almost twice that of the normal case. By that time the worm has spread to less than 10% of the vulnerable population (see Figure 2.(b)). This observation is crucial for early detection because it provides containment strategies more time to be effective. The same experiment was conducted assuming that only 75% of the mobile population is vulnerable. As Figure 2.(a) shows, the difference in moonwalk lengths is still significant.

More importantly, the results from Figure 2 suggest that a simple threshold detector could alert network operators about the presence of a mobile worm at its early stage of infection. Briefly, such a detector periodically calculates the moonwalk's length and keeps a running average of this length (*e.g.* using an exponentially-weighted moving average). When the difference between the current length and the long-term average passes a threshold, the detector raises

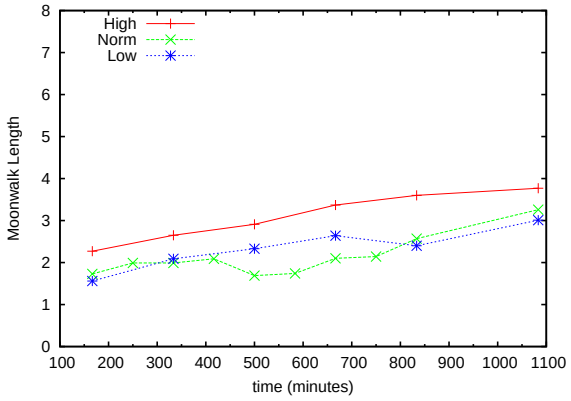

Figure 3: Average moonwalk length for a network with different volumes of normal traffic. The curve labelled 'High' corresponds to double the volume of traffic in 'Norm', while the curve labelled 'Low' represents a scenario in which the traffic is halved.

an alarm about an actively spreading worm. Since user traffic can vary over time (*e.g.* traffic volume during the day is usually higher than during the night), we conducted a simulation wherein we measure the average moonwalk length under varying traffic volume. Figure 3 represents the average moonwalk length when the normal background traffic is halved and doubled. As can be seen, while there is a small increase in the average moonwalk length, it is not as marked as in the case when a worm is present.

4. WORM IDENTIFICATION

The modified moonwalk algorithm from the previous section can detect the presence of a mobile worm. We now show how it can be modified to identify the first infected node (also known as *patient zero*) as well as other nodes infected during the infection's onset. Depending on the speed of detection, the identities of these nodes can be used to thwart the infection, for example by blocking traffic from those nodes and inspecting their traffic to generate attack signatures [9, 14].

At the same time, it is generally impossible to pinpoint the patient zero(s) of any infection using a purely flow-centric approach such as moonwalks. To see this, consider a scenario in which the patient zero was contacted by a benign node prior to the start of the infection. It is difficult to infer which of these two nodes is in fact the true patient zero without inspecting the contents of the flow between these nodes or the nodes themselves. What the moonwalk algorithm can achieve is to considerably reduce the number of nodes that should be inspected in order to reveal the origins of the infection.

Specifically, the algorithm identifies a small set of candidate infection trees, one of which is the true infection tree. In this context we define the infection tree as the graph induced by the worm node infection sequence. The first step in this process is to identify each of these trees' roots. To do so, we modify the moonwalk algorithm in three key aspects. First, we include a stopping condition to the moonwalk, a parameter T_s which denotes the estimated time when the infection started. We then halt every moonwalk when it

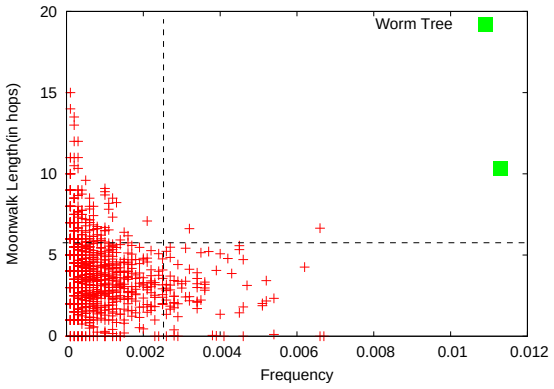

Figure 4: The scatter plot of walk length versus root node frequency. The square dot indicates the actual patient zero.

proceeds past T_s. This parameter can be estimated using the detection algorithm described in Section 3. Specifically, we showed that just before the infection enters its exponential increase phase, the average path length departs from the normal average. Then, if the worm is detected at time T_d, we set $T_s = T_d - \Delta T$. The value of ΔT depends on the mobility pattern and worm characteristics. In general, given a specific network and mobility model, ΔT should be set to the amount of time required for an infection to propagate to a popular domain from any wireless domain in the network. Second, in addition to edge frequencies, we record the frequencies with which each root node (*i.e.* alleged patient zero) appears in the moonwalks as well as the average walk lengths associated with each of these root nodes. Finally, we start each walk randomly, but only from nodes within the top p% popular domains. The rationale for this choice is based on the observation that a mobile infection in its initial phase moves towards domains of high popularity [12]. Therefore it is more likely, that during the early stages of the infection a malicious flow will be encountered in popular domains. We set $p = 25\%$ to maximize the probability that at least some of the moonwalks will follow a backwards path on the infection tree.

Once we carry out the random moonwalks, we draw the scatter plot of the root node frequency versus the walk length. The outliers in this scatter plot, that is the nodes with high frequencies and long walk lengths, are the possible roots of the infection trees. The intuition behind this approach is two-fold: **(a)** The frequency of the actual patient zero is high because worms tend to form tree-like structures and therefore multiple reverse paths lead to that node. **(b)** Unlike worms, non-malicious node contacts do not tend to form long paths.

We evaluate the performance of the algorithm using the simulation setup presented earlier. Figure 4 illustrates the scatter plot from the output of 10,000 moonwalks on 2.4×10^5 flows, with ΔT set to five hours. We use a simple filtering algorithm for identifying the outliers from this scatter plot. Points having walk length and frequency greater than 90% of the rest are chosen as outliers. The dashed horizontal and vertical lines in Figure 4 represent these 90^{th} percentiles. The points in the upper right quadrant of the graph are the roots of the candidate infection trees. The actual patient zero is also shown in Figure 4 with a square dot. While more sophisticated outlier detection algorithms could be used, we found in practice that this simple approach produces a small list of candidates that always includes the actual patient zero.

Starting from each of the candidate patient zeros, we reconstruct the candidate infection trees using all the edges that were traversed during the moonwalk phase. This is done using a simple breadth first search (BFS) discovery. Figure 5 presents the results of this traversal for three of the candidates from Figure 4. The nodes in these trees need to be further inspected for signs of infection. While the actual inspection method is out of the scope of this work, we can show that only a small percentage of nodes must be inspected. For example, if we traverse all tree nodes up to depth three, then $\sim$ 2% of the total population must be inspected. As an aside, the actual infection tree shown in Figure 5 was the one rooted at node 5344 and all the nodes in that tree were actual infected nodes. This result is encouraging because it indicates that the number of nodes falsely identified as active spreaders is rather low.

Last, we investigate the effect of different T_s (estimated infection start time) values on the outcomes of the algorithm. In our experiments T_s is set to the detection time minus ΔT (five hours). In most cases, we noticed that this is a slight underestimation of the actual infection onset. The problem with underestimation is that the true patient zero could have been contacted by other nodes in this time, thereby reducing the frequency of these nodes in the moonwalk. Nonetheless, the proposed algorithm is still effective, as the nodes which contacted the true patient zero still have relatively high frequencies and long path lengths in lieu of the patient zero having a high frequency and path length. The downside is that as the estimation error increases, the number of nodes that need to be inspected also increases. In our experiments, we noticed that even if the actual start time is underestimated by about 10,000 seconds ($\sim$2.8 hours), we needed to inspect at most 5% of the total node population.

4.1 Discussion

We have shown that the infection tree has both a high patient zero occurrence frequency and a long walk length. However, as the volume of normal traffic increases, it adds more noise to the selection algorithm. In other words, the normal traffic starts forming trees with lengths comparable to those of infection trees. As a result, the number of candidate trees to inspect increases. In the extreme case, the infection tree could remain 'hidden' within the volume of normal traffic.

We investigate the effect of the normal traffic volume by running the proposed identification algorithm on networks with increasingly higher levels of normal traffic. To do so, we keep the Lognormal distribution of the inter-arrival time of flows presented in Section 2.2, but decrease the mean inter-arrival time thus generating increasing levels of normal traffic. As Figure 6 illustrates, the percentage of mobile nodes that should be investigated for signs of infection increases as hosts spawn flows faster. Nonetheless, two encouraging observations can be made. First, the algorithm continues to identify the infection tree as the volume of normal traffic increases. Second, a decrease in the inter-arrival time by three orders of magnitude increases the number of nodes that must be inspected only by sixfold. Even when

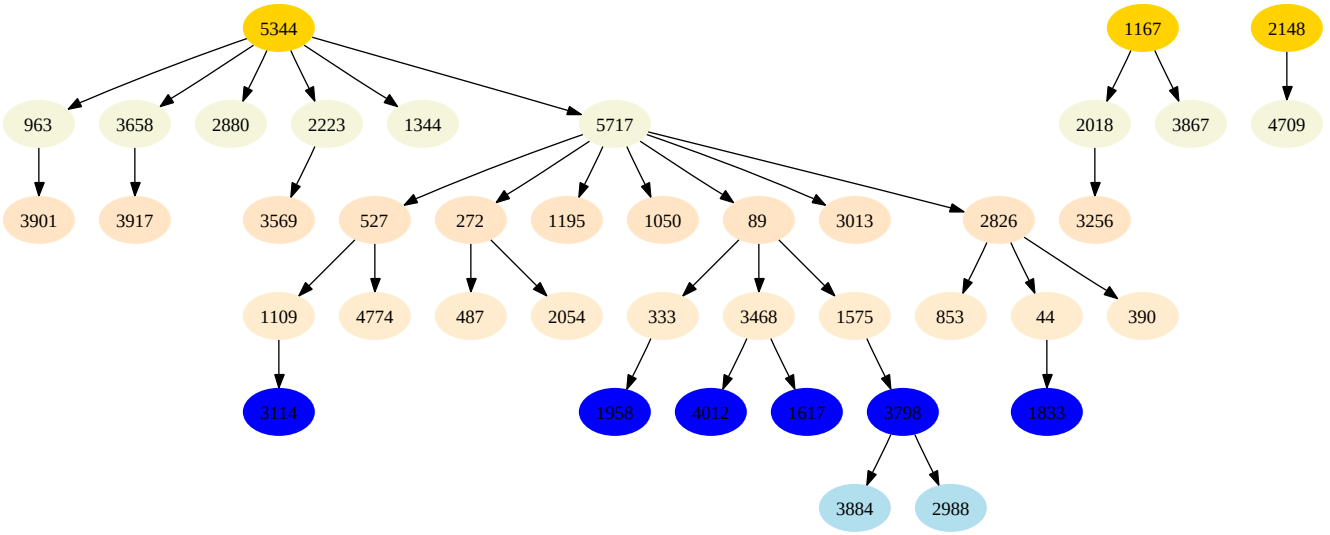

Figure 5: Candidate infection trees reconstructed using a BFS search. The tree rooted at 5344 is the actual infection tree. All nodes in this tree were indeed infected by the worm. The trees rooted at 1167 and 2148 are benign. A directed edge between node X and Y indicates that X initiated at least one flow to Y.

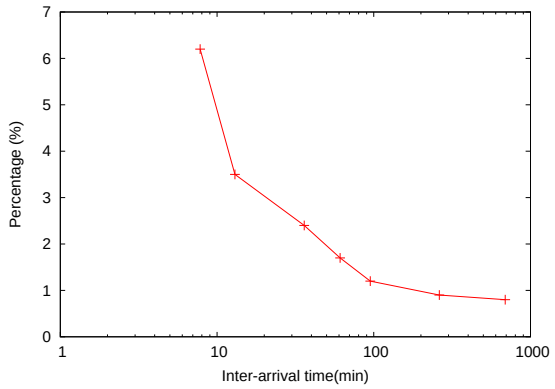

Figure 6: Percentage of mobiles nodes that need to be inspected for signs of infection as a function of the normal traffic intensity.

the average flow inter-arrival time is roughly ten minutes—a high value for intra domain traffic in wireless networks—the algorithm needs to inspect only 6% of the overall node population.

Finally, we briefly address a few issues on why the deployability of the proposed framework could be practical with reference to the latency in real time log collection, diverse background traffic and the size of connection logs. An in depth study is deferred for future work. As noted earlier, we assume that flow records are either aggregated at a centralized database, or available through a federated database (as proposed in NFA [18]). As seen from the figure 2, the detection algorithm can bear upto a latency of an hour, while still providing early detection. With regards to other types of background traffic, P2P traffic also tends to form long paths. However, they don't cause large changes in the path lengths. Since our detection algorithm concentrates on the changes in the path lengths, worm and normal P2P traffic can be differentiated. With regards to the amount of space required to store the flow headers of the intra-domain traffic – assuming that the source host ID, destination host ID, domain ID, start and end times of a flow require four bytes each, a flow record can be described in 20 bytes. In this case, even if we assume that nodes initiate new connections at a rapid pace of one flow every minute, the storage space required for the simulated network of 6,000 nodes is a modest 165 MB/day. Moreover, we expect the storage space to increase slowly as the number of hosts increases because the host contact graph is sparse.

5. RELATED WORK

The threat of mobile infections was first discussed by Anderson *et al.* [2]. Sarat *et al.* derived the speed of mobile worms through analysis and simulations [12], while more recently attacks against metro-area wireless networks were discussed by Akritidis *et al.* [1].

Our work is inspired by the work of Xie *et al.* on random moonwalks [17, 18]. However, that technique is primarily a post-mortem tool for identifying infected nodes, in the context of Internet worms, using host contact graphs. As we showed in Section 2.2, the effectiveness of the standard moonwalk method decreases rapidly as worms become more stealthy and infections are carried across domains by mobile nodes. We address the limitations of the original approach by exploring different heuristics such as moonwalk length and node occurrence frequency and show that moonwalks can in fact be used as a tool for the detection and origin identification of mobile worms.

Origin identification has been studied in the context of Internet worms. For example, Kumar *et al.* presented a forensic analysis of the Witty Worm [15], by reverse engineering the random number generator used by the worm [10]. In contrast, our technique is flow based and thus worm agnostic. Recently, there has been a body of work on securing

enterprise networks [6,7]. While the network environment is orthogonal to the one used in this paper, we believe that the moonwalk technique presented herein can play a role within such centralized architectures to detect and provide forensic analysis of malicious activities.

6. SUMMARY AND FUTURE WORK

This paper presents mechanisms to detect the existence and to identify the evolution of worms spreading through a collection of wireless domains, carried by the physical movement of mobile hosts. The proposed approach extends the existing framework of random moonwalks by focusing on the combination of moonwalk lengths and node frequencies to detect the existence of a stealthy worm and determine the identities of the infection's initial victims.

While we evaluate these algorithms in the context of mobile networks, we believe that they are also applicable to other worm scenarios. Because moonwalks essentially cull out worm edges in the presence of noisy background traffic, we believe them to be robust in the presence of missing traffic or in a distributed scenario in which some domains are non-cooperative. Further investigation of these claims is part of our future work.

Acknowledgments

We gratefully acknowledge the use of trace data from the CRAWDAD archive at Dartmouth College. We thank Fabian Monrose, Razvan Musaloiu-E., and Moheeb Abu Rajab for their suggestions. Brian Hoffman helped immensely in collecting the intranet data traces. We also extend our gratitude to the anonymous reviewers for their insightful comments and feedback.

This work is supported in part by the National Science Foundation through grant CNS-0627611.

7. REFERENCES

[1] P. Akritidis, W.Y. Chin, V.T. Lam, S. Sidiroglou, and K.G. Anagnostakis. Proximity Breeds Danger: Emerging Threats in Metro-area Wireless Networks. In *Proceedings of the 16th USENIX Security Symposium*, 2007.

[2] Everett Anderson, Kevin Eustice, Shane Markstrum, Mark Hansen, and Peter Reiher. Mobile contagion: Simulation of infection and defense. In *Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation (PADS)*, pages 80–87, Washington, DC, USA, 2005. IEEE Computer Society.

[3] Michael Bailey, Evan Cooke, Farnam Jahanian, Jose Nazario, and David Watson. Internet motion sensor: A distributed blackhole monitoring system. In *Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS)*, 2005.

[4] John Bethencourt, Jason Franklin, and Mary Vernon. Mapping Internet Sensors with Probe Response Attacks. In *Proceedings of the* 14th *USENIX Security Symposium*, pages 193–212, August 2005.

[5] Felix Campos, Merkourios Karaliopoulos, Maria Papadopouli, and Haipeng Shen. Spatio-Temporal Modeling of Traffic Workload in a Campus WLAN. In *Proceedings of the Second Annual International Wireless Internet Conference*, Boston, USA, 2006.

[6] Martin Casado, Michael Freedman, Justin Pettit, Jianying Luo, Nick McKeowon, and Scott Shenker. ETHANE: Taking Control of the Enterprise. In *Proceedings of ACM SIGCOMM*, 2007.

[7] Martin Casado, Tal Garfinkel, Michael Freedman, Aditya Akella, Dan Boneh, Nick McKeowon, and Scott Shenker. SANE: A Protection Architecture for Enterprise Networks. In *Proceedings of the 15th Usenix Security Symposium*, August 2006.

[8] CRAWDAD: A community resource for archiving wireless data at Dartmouth. Available from: `http://crawdad.cs.dartmouth.edu/dartmouth/campus`.

[9] A.-H. Kim and B. Karp. Autograph: Toward Automated, Distributed Worm Signature Detection. In *Proceedings of the 13th Usenix Security Symposium (Security 2004)*, 2004.

[10] Abhishek Kumar, Vern Paxson, and Nicholas Weaver. Exploiting Underlying Structure for Detailed Reconstruction of an Internet-scale Event. In *Proceedings of Usenix* 1st *Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2005.

[11] Moheeb Abu Rajab, Fabian Monrose, and Andreas Terzis. Fast and Evasive Attacks: Highlighting the challenges ahead. In *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID)*, September 2006.

[12] Sandeep Sarat and Andreas Terzis. On Using Mobility to Propagate Malware. In *Proceedings of the 5th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Network (WiOpt)*, April 2007.

[13] Stuart E. Schechter, Jaeyeon Jung, and Arthur W. Berger. Fast detection of scanning worm infections. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2004.

[14] Sumeet Singh, Cristian Estan, George Varghese, and Stefan Savage. Automated Worm Fingerprinting. In *Proceedings of the 6th ACM/USENIX Symposium on Operating System Design and Implementation (OSDI)*, 2004.

[15] W32.Witty.Worm, March 2004. Available from: `http://securityresponse.symantec.com/avcenter/venc/data/w32.witty.worm.html`.

[16] D. Whyte, E. Kranakis, and P. Van OorSchot. ARP-Based Detection of Scanning Worms within an Enterprise Network. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2005.

[17] Yinglian Xie, Vyas Sekar, David A. Maltz, Michael K. Reiter, and Hui Zhang. Worm Origin Identification Using Random Moonwalks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 242–256, May 2005.

[18] Yinglian Xie, Vyas Sekar, Michael K. Reiter, and Hui Zhang. Forensic Analysis for Epidemic Attacks in Federated Networks. In *Proceedings of the IEEE International Conference on Network Protocols*, October 2006.

[19] Zotob Causes Carnage in Corporate Networks. Available from: `http://www.netfastusa.com/xq/asp/id.1338/p.5-6-1/qx/PressRelease_view.htm`.