Is Host-Based Anomaly Detection + Temporal Correlation = Worm Causality?

Vyas Sekar, Yinglian Xie, Michael K. Reiter, Hui Zhang

March 6, 2007
CMU-CS-07-112

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

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.

Keywords: Worm, Attack Causality, Epidemic Attacks, Forensics

Abstract

Epidemic-spreading attacks (e.g., worm and botnet propagation) have a natural notion of attack causality – a single network flow causes a victim host to get infected and subsequently spread the attack. This paper is motivated by a simple question regarding the diagnosis of such attacks – is it possible to establish attack-causality through network-level monitoring, without relying on signatures and attack-specific properties? Using the observation that communication patterns of normal hosts are sparse, we posit the hypothesis that it is feasible to uncover attack causality through a combination of host-based anomaly detection and temporal correlation of network events. The contribution of this paper is a systematic exploration of this hypothesis over the spectrum of attack properties and system design options. Our analysis, trace-driven experiments, and real prototype based study suggest that it is feasible to establish attack causality accurately using anomaly detection and temporal event correlation in enterprise network environments with tens of thousands of hosts.

1 Introduction

Epidemic-spreading attacks (such as worm and botnet propagation) present a significant threat to the security of networks. Understanding and defending against such self-propagating attacks in an automated fashion is a challenging task. For each host infected by these attacks, the notion of attack causality[1] arises naturally. There is a single traffic event that causes this host to become compromised and spread the attack further. In this paper, we seek to understand if it is feasible to establish attack causality, i.e., provide the ability to pinpoint the *causal* flow which caused a vulnerable host to get infected. As attacks become increasingly sophisticated (e.g., changing payloads to evade signature-based detection, varying port numbers to evade firewall rules), we are particularly interested in techniques that do not depend on attack-specific properties such as signatures, and do not require prior in-depth understanding of attacks (e.g., the scanning strategies and scanning rates).

Our work goes beyond traditional attack detection in that we strive to provide capabilities that can establish the causal chain of events to describe how an attack unfolded across the network. This has tremendous value for attack forensics [25, 8, 14] and for guiding attack-signature generation (e.g., [6]). However, current approaches for establishing attack causality require fine-grained host-level analysis (e.g., [12, 21]) and often require updating each end-host in a network with new software capabilities. We explore the viability of an alternative lightweight *network-based* approach for establishing attack causality, which does not require fine-grained packet payload analysis, and can be implemented without modifying end-hosts.

To make our problem more concrete, we focus on enterprise environments where administrators can audit traffic events such as flow or packet headers for every end-host within their network. Even in this context, establishing causality in an attack-agnostic fashion is challenging, since the problem of inferring the *intent* of a particular traffic event without understanding the contents, application handlers, and end-host configurations appears intractable.

However, there are properties of real-world traffic patterns that can help in establishing attack causality using *network-level* monitoring alone. The communication patterns of individual hosts tends to be *sparse* – normal (i.e., non-infected) hosts tend to communicate only with a small set of hosts in the network on a regular basis, and the set of hosts contacted does not grow rapidly with time. This suggests that detecting infected hosts using coarse-grained anomaly detection metrics (such as the number of distinct destinations contacted) and without relying on attack-specific properties is feasible. Also, a majority of enterprise network traffic and a significant portion of Internet traffic is based on a client-server model. Thus the number of incoming connections to a client host is small. This implies that once we identify an infected host there are only a small number of incoming connections that serve as candidate flows for examining attack causality. Based on these observations, we hypothesize that it is feasible to establish worm causality, by combining host-based [2] anomaly detection with temporal correlation across infection events.

[1] The notion of attack causality is different from the notion of causality used in distributed systems [9]. We are interested in the exact flow that caused the host to be infected, and not on the temporal ordering among network events.

[2] This is referred to as a host-based detection system only for the reason that we want to detect anomalies for each host in the network. The detection only depends on observing coarse-grained network behavior of each host, and need

We present a systematic exploration of this hypothesis. First, we outline the design-space of the host-based anomaly detection and temporal-correlation in Section 5. Second, we use a three-fold evaluation methodology: (1) We use an analytical study with a simplified network traffic model (Section 4). This sheds light on the intuition behind the hypothesis and the factors affecting performance. (2) We present trace-driven evaluations (Section 6) using traces from a large university network (with over 16000 active hosts) where we vary both the spectrum of attack properties and the space of design options. Our evaluation shows that for common attack models that we observe today, we can establish attack causality with more than 95% accuracy, using network-level monitoring alone. We find most sources of inaccuracy arise from background scanning activity and server-driven behavior, which are easy to automatically cull out. For stealthy attacks that mimic normal traffic patterns or employ an *incubation* strategy, the approach is still promising but may require additional information regarding attack parameters or knowledge of the background traffic. We suggest novel mechanisms to automatically infer these features (Section 5.3 and 5.4), which are also easy to implement in practice. (3) We implement a prototype system and test it using a 25-day long trace from the same large university network (Section 7), and observe that the overhead of such a system is low even for large enterprise networks with tens of thousands of hosts.

These results provide the basis for a practical scheme for establishing attack causality, using only coarse-grained network-level monitoring, without relying on prior knowledge of attack-specific properties. This has positive implications for attack-defense, and such an approach will be immediately applicable in enterprise settings, with potential extensions for wide-area networks.

2 Related Work

Detection and correlation are recurrent themes in the intrusion detection and anomaly detection literature. Numerous research efforts focus on designing effective methods for detecting worm outbreaks and infected hosts (e.g., [24, 16, 22]). While these solutions do not address the notion of attack causality they provide anomaly detection capabilities which can be used in our framework. Recent work [7] utilizes both the temporal and spatial correlation of events for attack detection. We present the hypothesis that by combining detection and temporal correlation it is feasible to establish attack causality.

Establishing causality among traffic events has been previously studied in the context of stepping-stone detection (e.g., [19, 27]). These techniques analyze similarity of traffic content across flows, or perform fine-grained inter-packet timing analysis to establish causality between traffic flows. Our approach does not require such fine-grained analysis, but instead depends on only coarse-grained detection and flow-level analysis. Recent work by Kannan et al. [5] attempts to uncover hidden causality among traffic events using statistical properties of traffic arrivals. Our work differs from their approach in two aspects. First, we focus specifically on epidemic-spreading attacks (e.g., worm and botnet propagation) where there is often a discernible change in the behavior of an infected host. Second, their approach shares some similarity with the stepping-stone detection literature in that they depend on assumptions regarding statistical properties of packet inter-arrival

not be co-located with each host. In the remainder of the paper we use the term host-based anomaly detection with the understanding that the detection system monitors the network activity of each host in the network.

times. Attacks which vary the incubation time of an infected host and delay the onset of attack activity can evade such techniques which depend on the statistical properties of attack traffic to be substantially different from normal traffic patterns. By leveraging the fundamental sparsity of inter-host communication patterns, our approach is robust across a wide spectrum of worm attacks.

The notion of correlating incoming and outgoing connections at anomalous hosts is a common theme between our work and some worm detection techniques (e.g., [1, 20, 4]). There are two key differences between these approaches and our design. First, these approaches focus on detection, whereas our focus is on establishing attack causality. Second, these approaches often rely on predefined attack-specific rules (e.g., attack signatures, port numbers). Our focus is on investigating the potential of utilizing host-based anomaly information with flow-level correlation constituting a more attack-agnostic approach.

Forensic analysis of Internet worms [25, 8, 14] has recently received attention. Xie et al. proposed a random moonwalk algorithm to detect the origin of an epidemic attack by identifying the initial causal flows in an attack [25]. Network telescopes have been suggested as an alternative approach to reconstruct worm attacks [8, 14]. Worm forensic analysis can benefit from a better understanding of attack causality, especially if causality can be established without relying on attack-specific information.

3 Establishing Worm Causality: An Overview

Figure 1 depicts a conceptual overview of how we can combine host-based anomaly detection with flow-level correlation to identify causal flows. The first step involves a host-based anomaly detector. This detector will identify anomalous events by auditing network traffic, and flag hosts associated with abnormal activities. The requirements we desire of such an anomaly detector are that it should: (1) operate on fairly coarse-grained network-level observations without depending on prior understanding of attacks, (2) have low false-negative and false-positive rates, and (3) provide timing information on when the anomalous behavior of the flagged hosts began.

This host-based anomaly information will be provided as input to the correlation module, which uses historical traffic data along with previously reported anomaly events to identify potential causal flows. The basic idea is to correlate the traffic flows between infected hosts and their approximate infection times for establishing causality. When the detection module outputs a new anomaly event indicating that a host h might be infected at time t, the correlation module queries the *traffic archive* to retrieve traffic flows originating from other anomalous hosts (this information can be obtained by querying the *anomaly history*) and incoming into host h before t. Among these flows, the correlation module selects a subset of flows to investigate as possible sources of attack causality.

Let us consider a simple example on how we can correlate network events, using the information from anomaly timestamps to identify causal flows. Suppose two hosts A and B are flagged as anomalies with infection times of t_a and t_b, respectively, with $t_b > t_a$. If during the time between t_a and t_b, A "talks" to B, and there are no other incoming flows into B between t_a and t_b, then we can consider this flow from A to B as a potential causal flow that caused B to get infected, based on the following rationale. Before the flow from A occurred, B was not anomalous. But B became

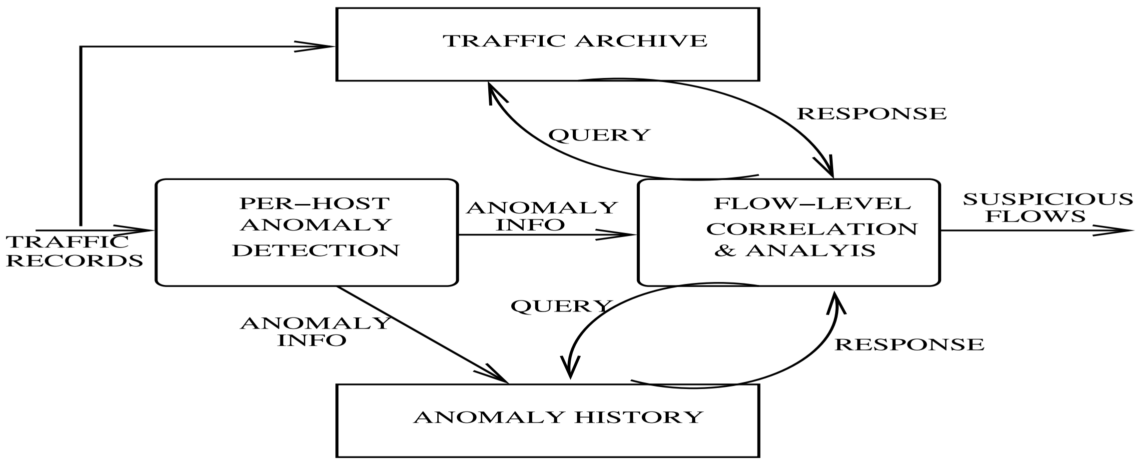


Figure 1: The host-based detection module identifies anomalous hosts. For each such anomaly, the correlation module analyzes candidate flows from the historical traffic archive which have the anomalous host as the destination.

anomalous after the flow occurred, and A is already known to be an anomalous host. It is therefore likely that this traffic event (A talking to B) *caused* the subsequent anomalous behavior on host B. In this example, we are merely using the timing information provided by the host-based anomaly detection system. It is possible to incorporate additional traffic features during this correlation step. For example, we can preclude known non-attack flows using port and server white-lists, and filter out such flows. Alternatively, we can automatically infer some properties of the attack (e.g., the destination port of the vulnerable service) and use these features to further refine the selection of candidate flows which we need to examine. Section 5.2 outlines the correlation step in greater detail.

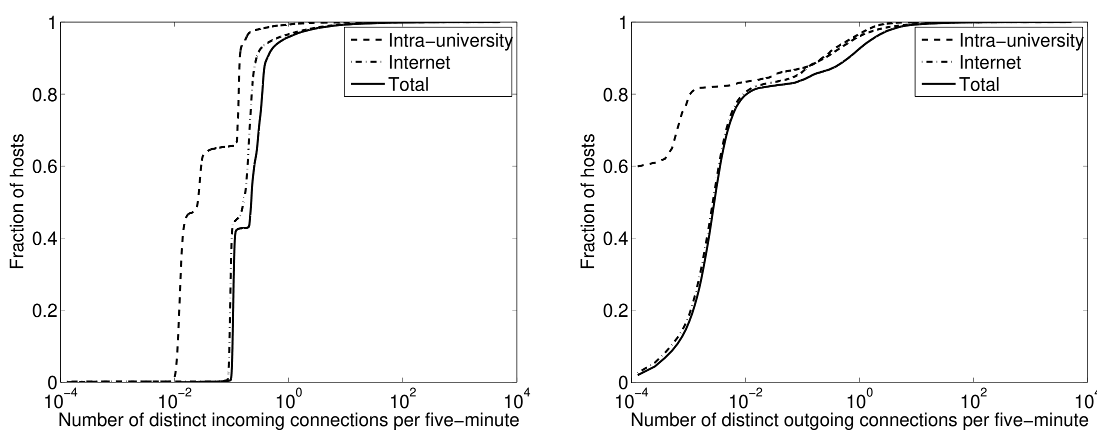
This approach builds on the intuition that communication patterns in network traffic are relatively sparse, both temporally and spatially. Temporal sparsity implies that the rates of communication of normal hosts tends to be relatively low. Spatial sparsity can be viewed along two dimensions. The first dimension is that the set of hosts that normal hosts communicate with tends to be stable over time. Earlier studies have shown that normal clients have significant locality in the set of hosts with which they communicate [23, 11, 17, 10]. The second dimension is that traffic patterns tend to exhibit predominantly client-server like behavior. This observation holds especially in enterprise environments; P2P applications are often restricted and the majority of connections are directed toward a small set of network servers [13]. This has favorable implications for our approach. First, we can utilize the sparsity of normal communication behavior to design host-based anomaly detection techniques that are independent of attack signatures, scanning rates, and other attack-specific properties. Second, spatial sparsity suggests that most normal clients will have few legitimate incoming connections. Thus once hosts have been flagged as having anomalous activity, we only need to look for *a small number* of incoming flows that precede the start of the anomaly. Therefore the likelihood that we will select the causal flow which actually caused the subsequent anomaly will be high.

While the above approach appears conceptually simple, several challenging questions remain. First, can we systematically confirm the high-level intuition behind this approach? Second, what is the design-space for the correlation module – e.g., how long into the history do we have to look back to select a candidate causal flow, how much performance improvement can we obtain by using additional traffic features? Third, how robust will this approach be across a spectrum of attacks? Last, can we realize such a system to operate in real-time with low overhead? We address these questions using analysis, trace-driven evaluations, and a real prototype based study.

4 Intuition

In this section we present some intuition behind our hypothesis, derived from two studies. First, we present a measurement study to confirm the sparsity in normal traffic patterns. Second, we present an analytical study under a simplified network model to reason about the approach outlined in the previous section.

4.1 Sparsity in normal network traffic



(a) Incoming connections (b) Outgoing connections

Figure 2: Measurement study of per-host behavior in a large university network

We present some quantitative justification regarding the sparsity of inter-host communication patterns. For this, we took a month-long trace (in Feb 2005) from a medium-sized university's core network. For each observed university host in the trace (there were in excess of 16000 unique active IP addresses), we find the number of distinct incoming and outgoing connections for every 5-minute interval, and average these values over the month. Here, a distinct connection refers to flows to/from distinct IP addresses. The connections are split into two categories, those within the university, and those which cross the border into the Internet. Figure 2(a) shows the distribution of the rates of traffic incoming to each of the identified hosts within the network, while Figure 2(b) shows similar results for the rates of outgoing traffic connections. We find that more than 90% of the hosts receive on average less than one distinct connection (both intra-university and Internet traffic together) over a five-minute interval. These results suggest that normal traffic patterns tend to be reasonably sparse, both in terms of traffic rates and the nature of inter-host communication patterns. Such trends are representative of enterprise environments, and similar observations have been echoed in several studies [11, 13, 23, 10].

4.2 Analytical Model

We use an analytical study with a simplified network and attack model to reason about the approach outlined in Section 3. We assume a ubiquitous monitoring infrastructure within a N-host network, where we can observe the behavior of all the N hosts. The attack is a worm-like attack spreading within the N-host network. We assume a discrete-time model of network traffic, in which every network flow (of the form $\langle src, dst, time \rangle$) has a length of one time unit, and each flow starts at the beginning of a time unit and finishes before the start of the next time unit.

To model the communication patterns of normal (non-infected) hosts, we assume each normal host initiates α concurrent outgoing flows per time unit. These normal traffic flows are a mix of client-server traffic and random destination traffic. Specifically, out of the α flows initiated by each host per time unit, a fraction U of the flows are to destinations selected uniformly at random. The remaining $1 - U$ fraction of flows are sent to a small number of servers (which are assumed to be immune to client vulnerabilities) in the network.

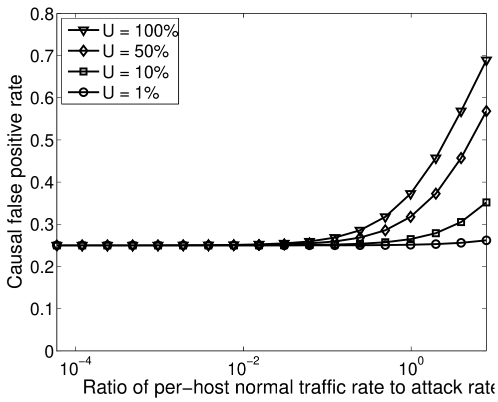
The attack is specified by the attack-rate and the fraction of vulnerable hosts (F). Once a host is infected, it starts malicious scanning at a rate of γ attack flows per time unit. Attacks use uniform random-scanning, where for each attack flow an infected host picks a random host from the N-host network. To simplify our analysis, we assume that attacks have zero *incubation time*, i.e., infected nodes start scanning as soon as they are infected. We will revisit the concept of incubation in Section 5.3.

Next, we assume there is a host-based anomaly detection system that reports infected hosts with zero false-negatives[3], but has a false-alarm rate β. Given β, the number of hosts that have been falsely flagged by the host-based anomaly detection system, i.e., the *host false-positive* is $HFP = \beta \times N$. For the purpose of our discussion, we assume that the time-line starts from 0, when the attack starts to spread. We assume that these host false alarms occur before the attack starts, noting that assumption will only cause the inaccuracy to be over-estimated[4].

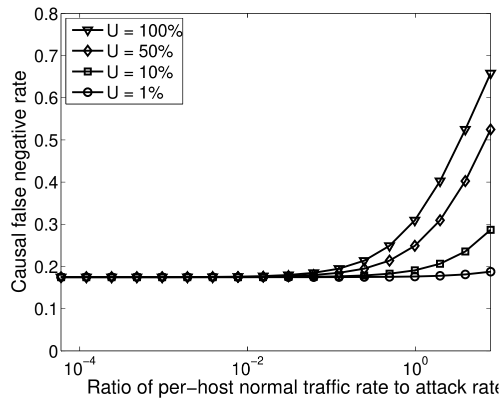
We are interested in the accuracy of identifying causal attack flows, in terms of the *causal false negative rate* (CFN) and the *causal false positive rate* (CFP). A causal false negative means that for an infected host h, we fail to identify the causal flow associated with the infection event. Suppose the number of causal flows that are missed is $misses$. The causal false negative rate will be the fraction of actual causal flows missed, i.e., $CFN = \frac{misses}{F \times N}$. The denominator $F \times N$ represents

[3]Using threshold-based detection we can design host-based detection systems with zero false-negatives. If the worm scan-rate is known, then we can set a threshold lower than the scan-rate to detect all infected hosts, i.e., with zero false-negative rate. When the scan-rate is not known, the multi-resolution approach [17] can provide zero false-negative rate over a spectrum of worm-rates.

[4]For each infected host h, we are going to look for candidate causal flows, i.e., an incoming connection into host h, from another host h' known to be anomalous prior to the connection. By assuming that all the host false-alarms occur before the start of the attack, we only increase the set of candidate flows that we need to examine. This assumption can only increase the inaccuracy.



(a) Causal False Positive



(b) Causal False Negative

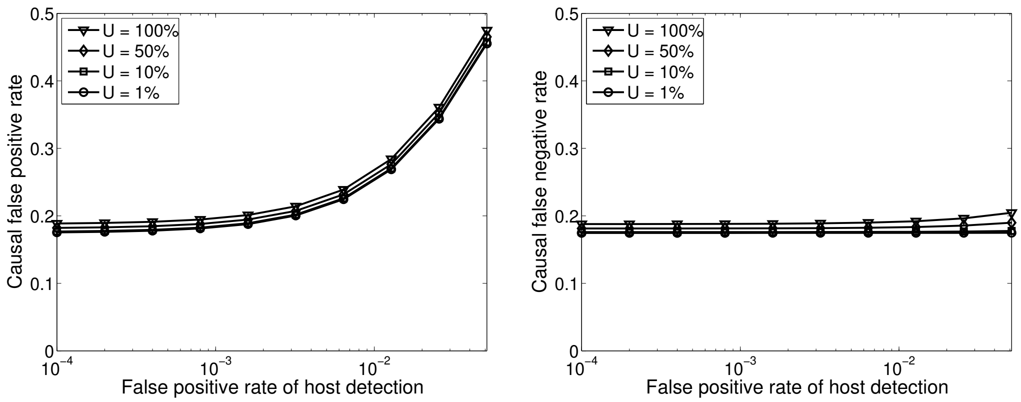
Figure 3: Varying background traffic rate, $\gamma = 5$, $F = 0.1$, $\beta = 0.05$

the number of infected hosts, and thus the total number of causal flows.

A causal false positive implies that a non-causal flow is reported as a suspicious flow. The causal false positive rate is then the fraction of non-causal flows among the set of flows returned. We assume that for each anomalous host (this includes the set of infected hosts and the false-alarms from the host-based anomaly detection system) we will be able to find at least one incoming candidate flow to return as the possible causal flow. Thus the total number of flows returned will be the number of anomalous hosts, i.e, $TotalReturned = NumInfected + NumFalseAlarms = F \times N + HFP$. To quantify the CFP, we need to identify the number of non-causal flows returned by the correlation module. There are two contributions to the set of causal false positives. The first contribution is the flows returned for the set of HFP false alarms. The second contribution will be non-causal flows which are falsely identified as causal flows for the infected hosts. Since the host-based anomaly detection component has no false-negatives, the number of causal flows which are missed ($misses$) is equal to the number of such non-causal flows reported, i.e., for every missed causal flow there is a corresponding contribution to the set of causal false positives. Thus, $CFP = \frac{misses+HFP}{TotalReturned} = \frac{misses+HFP}{F \times N+HFP}$.

Our task then is to quantify $misses$. Let us consider an infected host h which has been flagged by the host-based anomaly detection system at time i since the start of the attack. In the correlation step, we are going to look back into the previous time unit from the infection time (since we assume the attack has zero incubation time). To simplify our analysis, we assume that out of the set of candidate flows within this preceding time unit, the correlation module reports one of the flows at random[5]. Let $C(i)$ denote the number of candidate non-causal flows that arrive at the host h at time i, and $n(i)$ denote the number of hosts infected before time i. Now, candidate flows at time i can arise only from sources that have already been flagged as anomalies by time $i-1$. These hosts are either (a) hosts infected by the attack, or (b) false-alarms from the host-based anomaly detection system. There are $n(i-1)$ hosts infected at time $i-1$. Since one of these $n(i-1)$ is involved

[5]This eliminates the difficulty in modeling the temporal ordering between infection flows with identical timestamps.



(a) Causal False Positive (b) Causal False Negative

Figure 4: Varying β the host false positive rate, $\gamma = 5$, $F = 0.1$, $\alpha = 0.5$

in the actual causal flow, we only need to consider the flows from the remaining $n(i-1)-1$ infected hosts. These infected hosts will contribute both attack flows and normal flows to the set of candidate flows incoming into host h. The contribution of the attack flows is $\frac{(n(i-1)-1)\times\gamma}{N}$, and the contribution of the non-attack flows from these hosts is $\frac{(n(i-1)-1)\times\alpha\times U)}{N}$. In addition to these infected hosts, there are also the flows from the host false-positives. There are HFP such hosts each contributing $\frac{\alpha\times U}{N}$ flows to the set of candidate flows. The number of candidate flows is:

$$C(i) = \underbrace{\frac{(n(i-1)-1)\times(\gamma+\alpha\times U)}{N}}_{From\ infected\ hosts} + \underbrace{\frac{HFP\times\alpha\times U}{N}}_{From\ false\ alarm\ hosts}$$

In each of the two contributing terms, the factor U determines the fraction of normal background traffic that is intended to random destinations. Since all the flow terms in the above equation are based on selecting the destination uniformly at random, the term N in the denominator indicates the number of flows which will be reach a particular host h. Given $C(i)$, the probability of picking a non-causal flow coming into host h at time i will be $\frac{C(i)}{1+C(i)}$. The denominator here represents the number of candidate flows from which we need to select (the actual causal flow and the $C(i)$ non-causal flows).

$n(i)$, the number of infected hosts at time i can be estimated using the epidemic spreading model [3, 18]. In this model, the number of newly infected hosts at time i is related to the number infected at time $i-1$ as follows:

$$newinfected(i) = \begin{cases} 1 & i = 0 \\ n(i-1)\left[\gamma\times\frac{(F\times N-n(i-1))}{N}\right] & i > 0 \end{cases}$$

The number of hosts infected at time i is simply $n(i) = n(i-1) + newinfected(i)$.

The number of $misses$ is related to C and n values as:

$$misses = \sum_i n(i)\times\frac{C(i)}{1+C(i)}$$

We proceed to examine how the CFP and CFN depend on the normal traffic parameters, and the accuracy of the host-based anomaly detection system. For this study, we fix the network size to be $N = 15000$ and the attack scan-rate $\gamma = 5$. Figure 3 shows that as we increase the rate α of normal traffic flows, the CFP and CFN increase as expected. When the normal traffic rate is significantly more than the attack-rate (the x-axis represents the ratio between the per-host normal traffic rate α and the per-host attack-rate γ) the accuracy is very low for higher values of U, but at lower U the performance is independent of the normal traffic rate. This provides the first intuition: if normal traffic patterns are sparse, in terms of the rates and communication patterns, then we can establish causality with high accuracy.

In Figure 4, we vary β, the false positive rate of the host-based anomaly detection system. Intuitively, as the number of false alarms from the anomaly detection component increases, the CFP and CFN will increase. The CFP does show a sharp increase as we increase β, since the contribution of the causal false-positives returned for host false-positives increases. However, as long as the β is relatively low (i.e., less than 0.1) the CFN is almost constant. We also notice that when U is very small (i.e., client-server traffic predominates), the effect of β on the CFN is reduced further. This provides the second intuition: as long as the host-based anomaly detection system has a low false alarm rate, we can obtain accurate causality information, i.e., the causal false negative rate is low.

5 Approach

5.1 Host-Based Anomaly Detection

Our approach requires a host-based anomaly detection system to detect infected hosts and report their approximate infection times. While our framework can accommodate many anomaly detectors, in this paper we focus on using threshold-based detection based on monitoring the number of unique destinations contacted by each host. The strength of such threshold-based detection is that (a) it is easy to implement, and (b) it does not depend on attack-specific features such as signatures and scanning strategies. However, traditional threshold-based detection suffers from an inflexibility in threshold-selection, i.e., the spectrum of attack-rates which can be detected is tightly coupled to the threshold value and the window-size selected. For example, if we choose a threshold of 100 distinct connections in 10 seconds, it cannot detect attacks which have a scan-rate less than 10 scans per second. The multi-resolution approach [17] offers a simple extension to threshold-based detection, in which the detection system can be designed to be robust across a wide spectrum of attack rates. We adopt this approach for host-based anomaly detection. This idea is based on the simple observation that due to locality in end-host communication patterns, the number of distinct destinations each host contacts grows as a concave function of the size of the time window (i.e., the second derivative with respect to the time window size is negative). By simultaneously using multiple threshold values, each applied at a different time resolution, we can detect a wide range of attack rates.

The procedure for host-based anomaly detection using a multi-resolution approach is outlined in Figure 5. The detection system first obtains the number of distinct destination addresses con-

MultiresolutionDetection(W, T, H, M)
// W is the set of time resolutions
// $T : W \to \mathbb{R}$ is the set of thresholds
// H is the set of hosts
// $M : H \times W \to \mathbb{R}$ is the set of measurements
for each host $h \in |H|$ **do**
  for each window $w \in |W|$ **do**
    // Check if it exceeds the threshold
    if ($M(h, w) > T(w)$) **then**
      // Report the host, timestamp, and resolution
      Flag $\langle h, currenttime, w \rangle$ as an anomaly

Figure 5: Multi-resolution detection

tacted by each host in the network using sliding windows of different sizes (in the resolution set W) to obtain per-host measurements. $T(w)$ represents the threshold for the number of unique destinations contacted as a function of the time window for each $w \in W$. For each host h, and each window size w, it checks if the measured value is greater than the detection threshold $T(w)$. A host's behavior is flagged as anomalous if its activity exceeds the corresponding threshold for any of the constituent resolutions. Each alarm raised by the system is a tuple of the form $\langle hostid, timestamp, w \rangle$, which means that $hostid$ exceeded the connection threshold for the time window of size w ending at $timestamp$. The detector outputs a set of per-host alarms, indicating the alarm time, and the corresponding time-resolution at which the host was flagged. This information will be subsequently used by the correlation module.

Threshold Selection[6]: First, we select a set of window-sizes for the multi-resolution approach. In our evaluations (Section 6) and prototype implementation (Section 7) we use window sizes ranging from 5 seconds to 300 seconds, with intermediate values of 10, 20, 60, 100, and 200 seconds. We then proceed to analyze historical traffic records of host communication patterns. From these historical datasets, we obtain the distribution of traffic rates across all hosts for each of the window-sizes of interest. For example, for a window-size of 100 seconds, we compute the number of distinct destinations contacted by each host in our network over all possible sliding windows of duration 100 seconds over the traffic history. Given these observations, we proceed to obtain statistical percentiles over the distributions for each window-size. We select the 99.5th percentile of the distribution for each of the windows from 5 seconds to 300 seconds. These values (the number of distinct destinations contacted over different window sizes) are used as the connection thresholds (i.e., $T(w)$) in the multi-resolution approach.

[6]This is a simpler threshold-selection method compared to Sekar et al. [17]

FINDCAUSALFLOW($h, alarmtime, A, F, lookback$)

// h is the host on which the anomaly is reported
// A is the set of previously reported anomaly
// $flow = \langle src, dst, stime, etime, sport, dport \rangle$
// *flowcheck* is a boolean function on flow attributes
Get $PotentialCandidates$ from the traffic flow archive
$\{f | f.dst = h, f.etime \geq alarmtime - lookback\}$
Sort $PotentialCandidates$ in increasing order of $stime$
for each *flow* $\in PotentialCandidates$ **do**
  if (*flowcheck*(*flow*) $= TRUE$) **then**
    // Is the source of flow already anomalous?
    if ($A[src].start <$ *flow*$.stime < A[src].end$) **then**
      Report *flow* as being causal for host h

Figure 6: Temporal correlation among network flows to identify potentially causal events

5.2 Correlation

Given the input from the the detection module, i.e., a list of anomalous hosts and the corresponding anomaly timestamps the correlation module analyzes their communication events and outputs a list of potentially causal flows. In order to do so, the correlation module has access to the set of archived traffic records for the monitored network and previously reported per-host anomalies. Each entry in the historical anomaly database has two timestamps, indicating the start and end times of the anomaly event. We allow for the fact that anomalies may have an end-time, i.e., possibly indicating when the host was patched or quarantined from the network. If the anomaly does not end, we assume the end time of the anomaly to be ∞.

Given the alarm reported on host h, the procedure (Figure 6) first retrieves a set of incoming flows that have been observed at host h, in the last *lookback* seconds[7]. This set is then sorted in increasing order of the flow start time. Among these flows, we look for the earliest flow such that, (1) the flow satisfies the flow-condition *flowcheck*; and (2) the flow was initiated within the interval when its source was detected as an anomalous host. For a purely timing-based correlation approach, the condition *flowcheck* always returns true. When we use additional conditions, for example to use whitelists (Section 5.4.1) or to specify a filter based on destination ports (Section 5.4.2), then *flowcheck* can be modified suitably to specify these conditions.

Deciding an appropriate value of *lookback* is tricky for two reasons. First, since host-based anomaly detection systems can have a detection latency, the time at which the anomaly is reported might be greater than the actual infection time. The second source of inaccuracy arises from the attack dimension. For naive attacks which start scanning as soon as they are infected, it seems

[7] The analysis in Section 4.2 assumes that the reported infection times are accurate. In reality this may not be the case. Hence, we adopt a conservative strategy of looking back a finite interval instead of looking for an immediately preceding flow.

INFERINCUBATIONOFATTACK

1 Identify the start and stop time of the attack.
Obtain the attack duration d.
2 Compute the scan-rate γ of the attack.
3 Infer the fraction (F) of the host-population infected.
4 Simulate *hitlist* worms fixing parameters
(scan-rate=γ, duration=d, fraction vulnerable=F).
Vary the incubation period parameter P.
5 Obtain the best-fit incubation period that provides
the best-fit to the observed worm growth.
// Sim is the simulated worm growth
// Obs is the observed worm growth
$\hat{P} = \mathrm{argmin}_P\ L1Norm(Sim('Hitlist', \gamma, d, F, P), Obs)$

Figure 7: Inferring the incubation parameter

appropriate to look for flows immediately near the time of the anomaly. However, attacks can evade this limited notion of causality by incorporating a longer latency between the infection time and the start of scanning activity. Thus attacks could employ an *incubation* strategy to delay the onset of discernible anomalous activity after getting infected.

If the $lookback$ window is too small then we may miss the actual causal flow (increasing the causal false negative rate CFN), either because attacks use a long incubation period or because the detection system reports alarms later than the actual infection time. On the other hand, if the $lookback$ window is too large, we run the risk of reporting unrelated network events as possible sources of attack causality (i.e., increasing the causal false positive rate CFP). We define two strategies in determining the $lookback$ value. One alternative is an *immediate lookback* approach, which searches for flows within a finite window from the time the anomaly is reported. In our implementation of the immediate lookback approach, we use a $lookback$ window which is equal to the time-resolution on which the anomaly is reported. The alternative *infinite lookback* searches for flows within an infinite window from the anomaly.

5.3 Inferring the incubation period

As we saw in the earlier discussion, attacks can employ a smart *incubation* strategy. A natural question is whether a $lookback$ window can be inferred in such cases to account for the incubation period of the attack. One alternative is to analyze the worm payload to identify the exact incubation strategy used. We suggest an alternative data-driven approach (Figure 7), which does not depend on such worm-payload analysis.

The high-level idea behind our approach is that we reverse-engineer the attack's incubation parameter using the information we have from our traffic trace and anomaly information. Now, there are several unknowns: the attack scan-rate, the attack duration, the attack's scanning strategy, the number of hosts vulnerable to the attack in the network, and the incubation parameter. Our approach is to try and estimate all the unknowns except the incubation parameter, and then use simulations to infer the incubation parameter which best matches the observed worm growth.

First, we infer the attack start and stop time by temporally clustering the per-host alarms, and finding the start and stop times of the largest such cluster. From this we estimate the duration of the attack. The scanning-rate of the attack can be approximated by the average outgoing traffic rate of the hosts that have been flagged within this largest cluster. Since the traffic rates of non-attack traffic are not that high, this will be a good estimate of the attack scanning-rate. The fraction of hosts vulnerable can be approximated by counting the number of distinct hosts that have been flagged by the host-based detection system.

Since we may not know the worm scanning strategy without a detailed understanding of the attack, in the simulations we use a *hitlist* scan worm. The rationale is that using an attack with a more effective scanning strategy can only give a overestimate of the actual incubation period. A hitlist worm, by definition has prior knowledge of the vulnerable population and thus has a very effective scanning strategy (it always picks only vulnerable hosts when selecting destinations to scan). In other words, by using a *hitlist* worm in our inference if we do err in our assumption of the scanning strategy, it will only cause us to infer an incubation period which is larger than the actual attack incubation period. If we had inferred an incubation period smaller than attack's actual incubation period, it would lead to a very high CFN since the $lookback$ interval would be too small. By inferring a slightly larger incubation period, we will not incur such false-negatives.

The last step is to identify the incubation period that best explains our observed sequence of per-host anomalies, i.e., a maximum likelihood estimate of the incubation period over a range of possible values. We use a error metric defined as the L-1 norm difference between observed infection spread (i.e., the number of hosts infected as a function of time) and simulated infection spread over $k = 5$ independent runs, and pick the incubation period with the smallest L-1 norm error.

5.4 Improving Timing-Based Correlation

Next, we describe two alternatives for specifying additional flow features that can be specified within the flow-condition $flowcheck$. These additional features may trade-off some of the attack-agnostic characteristics of our basic approach. Nevertheless, they are practical solutions that can benefit system administrators to prune their search space, and it is prudent to examine these as viable enhancements.

5.4.1 Whitelisting

Rule-based whitelists are commonly used in IDSes to rule out known causes of anomalies to reduce the effective false alarm rate. IDSes such as Snort [15] typically provide rules to allow servers and server-connections on specific ports to be whitelisted. Proceeding on similar lines, we use a server and port-based whitelisting option. In addition we can also allow for known non-causal connections to be removed from consideration. These may include administrative scans or "failed" connections from background scanning activity [26]. Our first enhancement to the basic temporal correlation procedure is the use of whitelists to rule out known non-attack flows, which may potentially appear as having some causal relationship, and flows which are unlikely to have causal impact on host behavior (e.g., failed connections).

5.4.2 Iterative Feature Extraction

A second enhancement, which can be implemented while operating in post-mortem analysis mode, is to identify dominant features of attack traffic. These features can then be used to iteratively refine the set of suspicious flows. In the iterative refinement process, the first iteration starts by performing the basic temporal correlation procedure in Figure 6, with the flow-condition $flowcheck$ always returning true. At the end of the i^{th} iteration, we identify dominant traffic features that are common to the suspicious flows. Specifically, we identify destination ports that contribute a substantial fraction of suspicious flows. For subsequent iterations, we restrict ourselves to traffic on these identified ports, by updating $flowcheck$ to specify the subset of ports which are interesting. The iterative process continues till the set of returned suspicious flows does not change substantially across iterations. If no dominant ports are identified the refinement ends in that iteration. Since most known attacks propagate using a small number of vulnerabilities and vulnerable services, the intuition is that this iterative refinement will reduce the number of causal false positives. This may also reduce causal false negatives by avoiding causal misses arising because the system returns a non-causal flow instead of the actual causal flow.

6 Trace-Driven Evaluation

We use a seven-hour traffic trace from the core of a large university network to evaluate our approach. The trace was collected on Feb 8, 2005 between 1-8 pm. It contained over 5.5 million flows with over 16000 active hosts. The trace contains flow-level [2] records similar to Netflow, with the added capability to provide indications of session directionality of each flow using TCP flags and timestamps (for UDP flows). We identify all intra-university flows (using network-prefix information) from each trace. These serve as representative background (i.e., non-attack) traffic for our analysis. We use synthetically introduced attack traffic records in the following evaluations. We repeated the experiments with several different trace snapshots, but only present the results from the Feb 8 trace since the results were similar. Broadly, we answer the following questions in this section:

- How does the accuracy of causality determination vary across the spectrum of attack parameters including the scanning rate, scanning strategy (e.g., random vs. hitlist vs. topological worms), and the extent of attack stealthiness (in terms of the incubation parameter)?
- How does the infinite lookback approach compare with the immediate lookback strategy? Do we need an accurate lookback parameter for causality determination?
- What are the main factors contributing to inaccuracy, i.e, what are the reasons for false positives and false negatives?

- Does the use of additional traffic features in the correlation step (iterative feature extraction and whitelists) improve performance? Can this additional information compensate for poor lookback information?

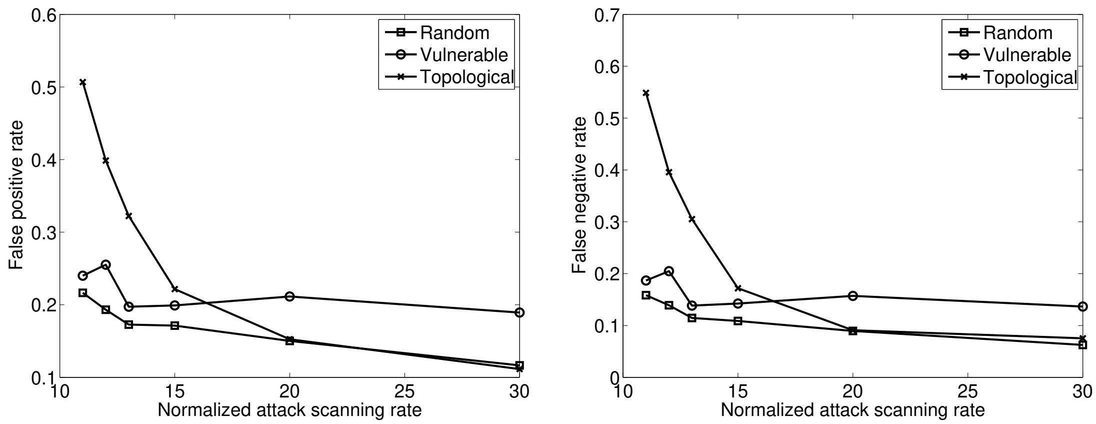
6.1 Varying Scan Model and Rate

We use three different worm scanning strategies: random-scan, hitlist, and topological-scan attacks. We also vary the range of worm-scanning rates (expressed in terms of a normalized rate, the ratio of the scanning rate of each infected host with respect to the mean per-host traffic rate we observe in normal traffic). The random scanning attack selects destination targets uniformly at random within the host address-space. The hitlist attack is assumed to have complete knowledge of the set of vulnerable hosts, and scans are restricted to these hosts. In the topological attack, the worm is assumed to have knowledge of each infected hosts normal communication patterns, and attack traffic mimics these normal communication patterns while selecting destinations to scan[8].

To evaluate the accuracy of detecting causal flows we use two natural metrics. The causal false positive rate (CFP) is defined as the fraction of identified flows which are non-causal, i.e., are not associated with attack causality in the synthetically introduced attack traffic. Some of these may have causal implications for other incidents unrelated to our synthetic attack traffic, but we report them as false positives to get a conservative overestimate. The causal false negative rate (CFN) is the fraction of causal flows that are not detected by our system. Figure 8 shows the causal detection accuracy in terms of the CFP and CFN using an immediate lookback approach. While the accuracy seems to be rate-independent for the random and hitlist worms (with the CFP around 20%) we find that the accuracy is poor for the topological scanning worm at low scan-rates. We also considered the alternative infinite lookback approach for these scenarios, but the performance is significantly worse (Figure 9), with the CFP and CFN exceeding 70%.

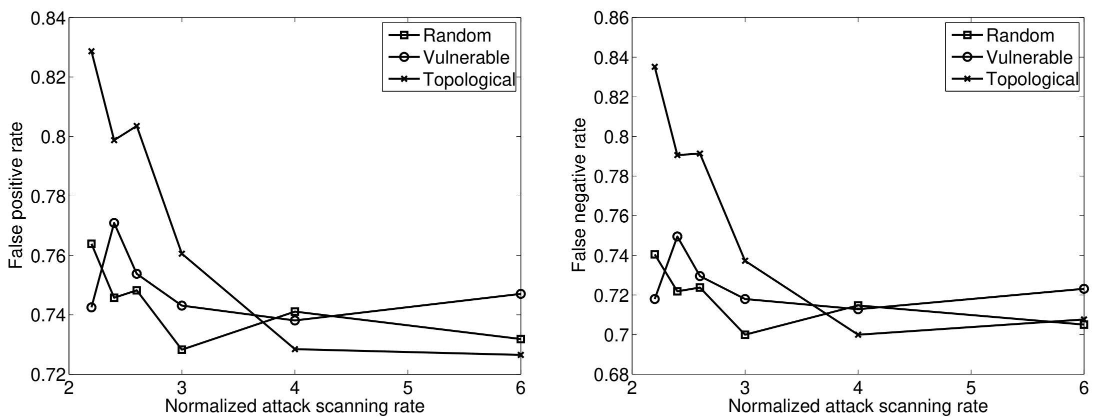
We are also interested in a breakdown of the types of misses among the causal false negatives (Figure 10). To further understand the nature of these misses, we categorize false negatives into several categories: 1. the anomaly detector failed to flag the source of the causal flow, 2. the anomaly detector failed to flag the destination of a causal flow, 3. the timing information provided by the detector for the source of the causal flow is incorrect (i.e., the reported infection time for the source is greater than the causal flow's timestamp), 4. the lookback period is insufficient to identify the causal flow, 5. the time at which the anomaly is reported on the destination of the causal flow is earlier than the causal flow, or 6. the system returned a candidate flow earlier than the actual causal flow even though the detection times are correct. The results for the hitlist attack are quite similar to the random-scan attack and are not presented in the remainder of our discussion. In both the random and topological attacks, we observe that the largest contributing factor is the presence of earlier candidate flows. For the topological worm, we find that at low scanning rates there is also a non-trivial contribution from source-timing errors, and even some misses from the anomaly detection system.

[8]In our specific implementation of the topological attack, an infected host picks scan-targets according to the following rule. With probability $p = 0.4$ it picks a destination it has already contacted, and with probability $p = 0.6$ it picks a random destination.



(a) Causal False Positive (b) Causal False Negative

Figure 8: Accuracy vs. worm scan-rate across different attack models with immediate lookback



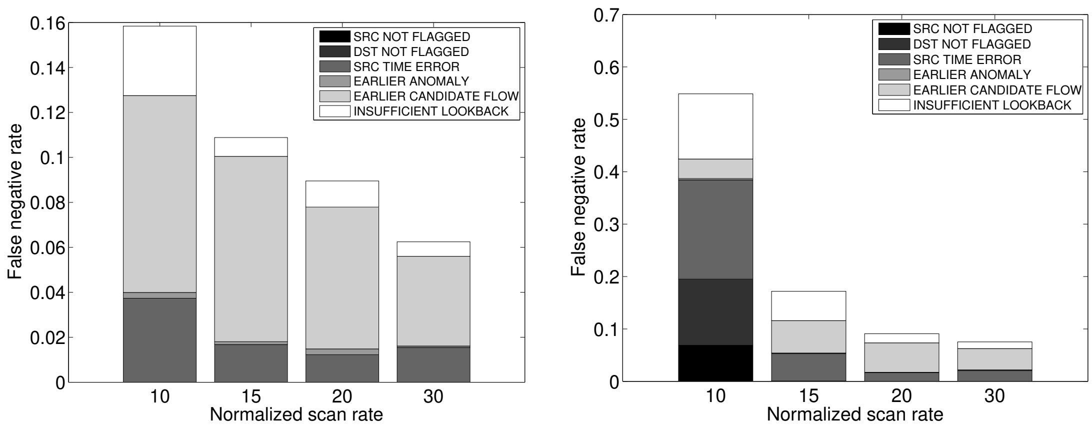
(a) False Positive (b) False Negative

Figure 9: Accuracy vs. worm scan-rate across different attack models with infinite lookback

A breakdown of the false negatives with the infinite lookback approach approach (Figure 11) also indicates that the vast majority of errors (more than 60%) arise because the system returns earlier candidate flows. Note that unlike the immediate lookback approach, we cannot have errors due to insufficient lookback, and thus we do not have to consider these sources of causal misses in the breakdown. The main disadvantage of an infinite lookback approach is that it may report traffic flows quite early in time unrelated to actual attack traffic as potential sources of attack causality.

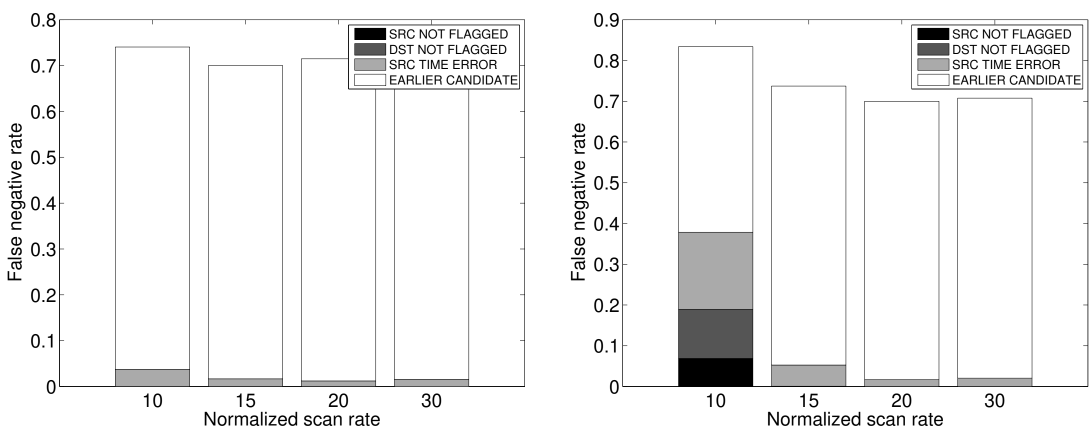
Analyzing the set of causal false positives (Figures 12 and 13) reveals that most of these flows have a few common features. They typically occur on a small number of destination ports, and arise from a small number of source addresses. The ports include common Windows-RPC services that are known to be associated with background scans (such as ports 135, 139, 445)[9], AFS servers (ports 7000-7002), DNS traffic (on port 53), some DHCP server traffic (port 67), and a small

[9]Since these scans are unrelated to the synthetically introduced attacks, we treat these flows as potential false-positives. Uncovering such potentially causal scanning traffic may not necessarily be construed as a false positive.



(a) Random (b) Topological

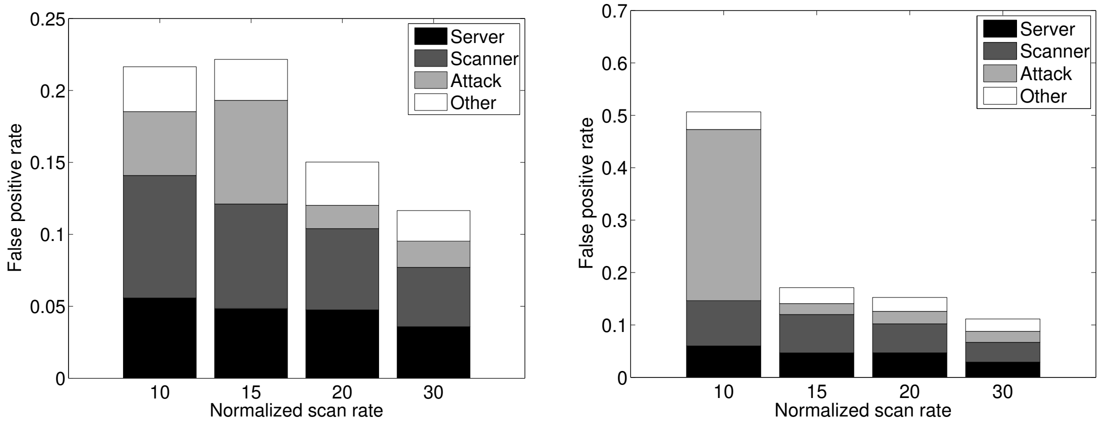
Figure 10: Breakdown of causal false negatives with immediate lookback



(a) Random (b) Topological

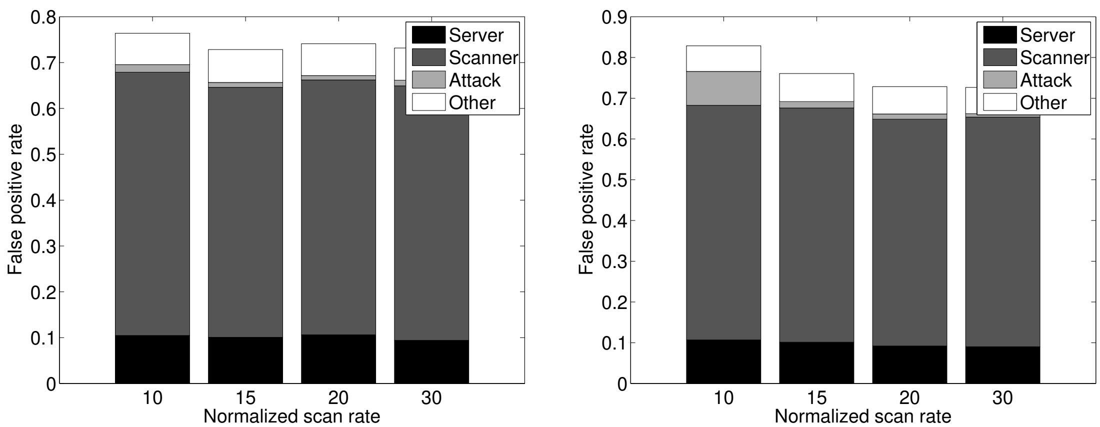
Figure 11: Breakdown of causal false negatives with infinite lookback

number of SNMP connections (on port 161). The fact that the causal false positives have a small number of known causes has favorable implications. By filtering out these known non-causal traffic flows, using whitelists and iterative feature extraction, we might be able to potentially improve the causal detection accuracy. The poor accuracy for low-rate topological worms also deserves further attention in understanding the sources of false-positives. We find in more than two-thirds of the causal false-positives were in fact attack flows (Figure 12 (b)). However, these attack flows are not causal but are repeated infection attempts, since the timing information was incorrect (i.e., the alarm is reported later than the actual infection event) for many of the hosts. Since the topological worm mimics the nature of normal communication patterns there are more errors from the anomaly detection system in this case.



(a) Random (b) Topological

Figure 12: Breakdown of causal false positives with immediate lookback

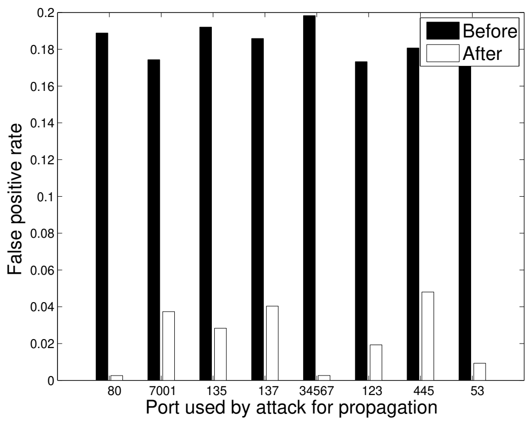


(a) Random (b) Topological

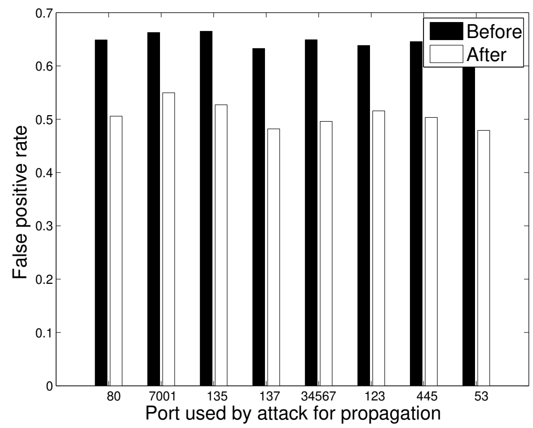
Figure 13: Breakdown of causal false positives with infinite lookback

6.2 Using attack feature identification

We focus on identifying the top k common destination ports from the set of suspicious flows (refer Section 5.4.2). Instead of selecting a fixed k, we identify ports which contribute more than 5% to the set of suspicious flows. Then, we iteratively refine the set of suspicious flows by repeating the correlation procedure (Figure 6) with the flow selection function $flowcheck$ updated each time to reflect the subset of destination ports that are of interest at the current iteration. The stopping condition for the iterative procedure is defined in terms of the similarity between the set of flows returned across successive iterations – if there is more than 95% overlap between the set of returned flows across two successive iterations we terminate the refinement at that stage. In our experiments we found that this process of refinement usually terminated in 3-4 iterations. Figure 14 depicts the improvement in the CFP by performing such an iterative attack feature refinement. Since the feature identification attempts to infer the destination port of the attack, we vary the port on



(a) Immediate Lookback

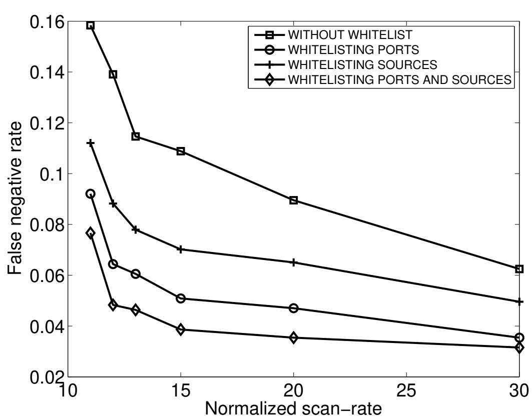


(b) Infinite lookback

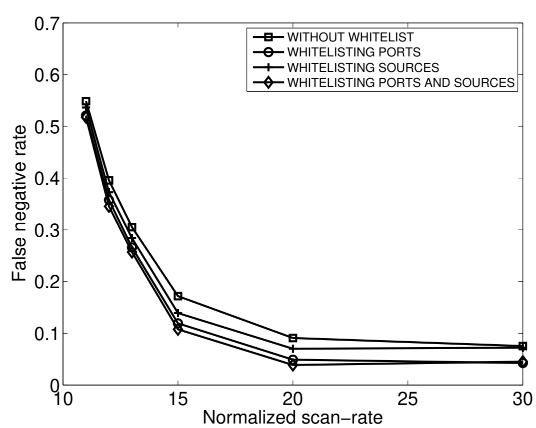
Figure 14: Improvement using feature extraction, varying the attack ports with a random-scan worm

which the attack can propagate. We observe that with the immediate lookback approach, there is a dramatic reduction in the CFP, independent of the port used by the attack. However, the improvement with the infinite lookback approach is less pronounced. The reason for this can be better understood by reflecting on the results from Figure 13. With an infinite lookback approach, causal false positives from scanners and servers constitute a majority of the flows identified after the first iteration. Given that we select ports which contribute more than 5% of suspicious flows, these flows do not get filtered out even after many iterations, since the starting configuration has a majority of non-attack flows.

6.3 Effect of Whitelists

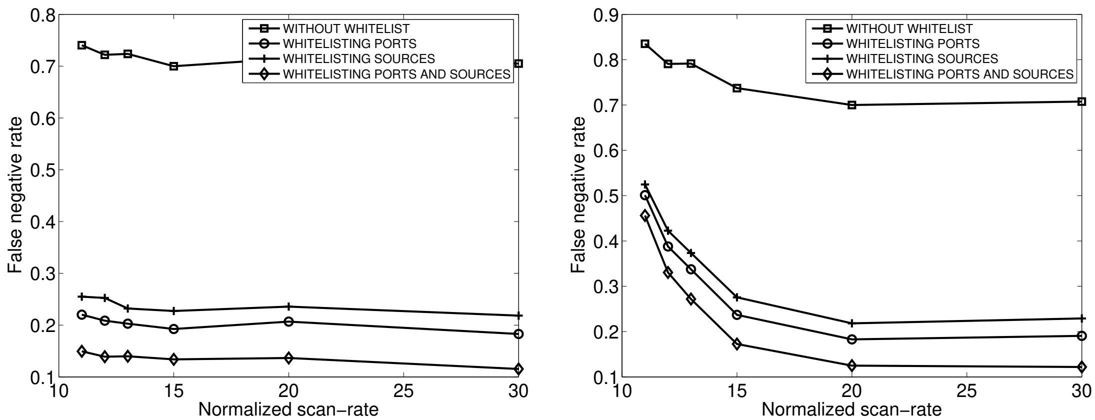


(a) Random



(b) Topological

Figure 15: Whitelists with immediate lookback



(a) Random (b) Topological

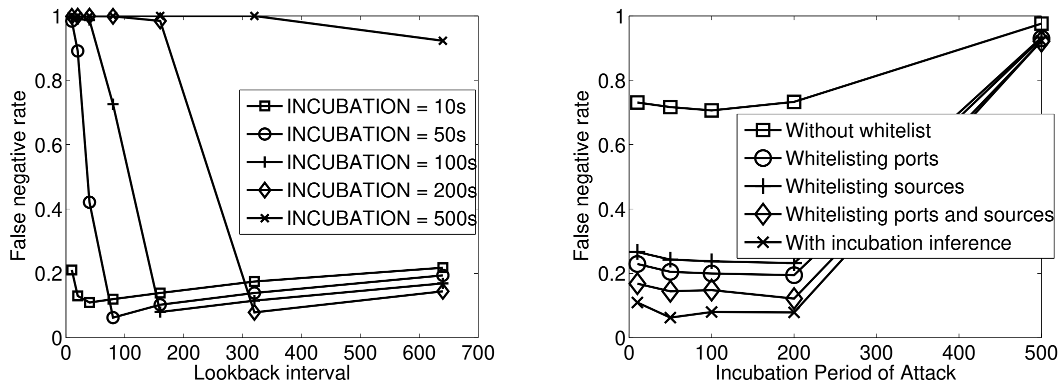
Figure 16: Whitelists with infinite lookback

We consider two-types of whitelists (refer Section 5.4.1): using destination port numbers and using source addresses. In each case, we use the top five contributing ports (sources) to the causal false-positives and consider the case when we ignore the flows involving these ports (sources) in the correlation step[10]. Recall (from Figures 12 and 13) that these traffic patterns typically correspond to servers and scanners. In Figure 15 we observe with the random scan attack, we get a two-fold improvement in the accuracy. With the topological attack, however, we notice that at lower scan-rates the performance improvement is not quite as significant. The reason is evident if we consider the results of Figure 12(b). At low scan-rates, a predominant number of causal false positives in the topological attack scenario are actually attack flows. Using whitelists will not alleviate this problem. The second question of interest is whether the additional whitelists aid in improving the performance of the infinite lookback strategy. This approach is appealing since it does not depend on accurate anomaly timestamps or lookback parameters. Figure 16 shows a promising result. Eliminating the known non-causal connections gives a three-fold improvement in the performance for both the topological and the random scan worms. We notice that even using a small white-list (top-five ports and hosts) enables us to obtain accuracy comparable to the immediate lookback approach. This suggests that having some additional information regarding background traffic patterns can lessen the dependence on accurate anomaly timing and lookback information.

6.4 Attacks with incubation

Attacks can evade the causality determination process by delaying the onset of anomalous behavior after the infection. In other words, once a host is infected, it can delay scanning activity for an *incubation* period. Thus the timing information derived from the host-based anomaly detection will cease to be a good estimate of the infection time, since it can only guide us to the start of the anomalous scanning activity. To observe how such an incubation strategy can defeat the causality determination step, we use a random-scanning worm (scan-rate to be 30 times the mean normal

[10]The top 5 ports were 137 (NetBIOS), 7000 (AFS), 445 (Windows RPC), 53 (DNS), 7003 (AFS)



(a) Accuracy vs. lookback (b) Infinite lookback

Figure 17: Accuracy of causality estimation for attacks with incubation strategy

traffic rate), and vary the incubation period. In our attack we assume that all infected hosts have the same incubation period. Figure 17 (a) confirms the intuition that knowledge of an ideal lookback parameter is indeed important to guarantee good causal detection accuracy. In each case, we also confirmed that the inferred incubation parameter (Section 5.3) is slightly larger than the optimal lookback value.

Interestingly, we find that for the attack with an incubation period of 500 seconds the performance is poor independent of the value of the lookback parameter. Investigating this particular case, we found that the anomaly detection did not flag many of the infected hosts. It turned out that the attack ended (our attack runs for a fixed duration of 2000 seconds) before many of the infected nodes began their scanning activity. Due to the relatively long incubation period, most of the nodes were infected relatively close to the end of the attack. Since these nodes also had a large incubation period, the onset of their scanning activity never actually occurred. In this case, anomaly detection was futile – there was no visible anomalous behavior for many of the infected hosts, and the ability to identify causal flows is weakened.

Since the infinite lookback strategy does not depend on accurate timing information, it can be potentially used for worms with incubation strategies without requiring the additional lookback inference step. We also found earlier (Figure 16) that the performance of the infinite lookback approach is dramatically boosted with the introduction of additional whitelist information. Building on the promise of this result, we evaluate the effect of using an infinite lookback approach for attacks with incubation strategies in Figure 17 (b). We compare the performance of this approach (infinite lookback enhanced with whitelist information) against the immediate lookback approach (using the incubation period inference). The performance difference between the two approaches is not significant. This result suggests that we can potentially remove the dependence on inferring an optimal lookback parameter, and thus be robust to possible attack evasion strategies (e.g., attacks can vary the incubation period across hosts to defeat the inference).

7 Feasibility Study

We have implemented a prototype of the causality inference system based on the design outlined in Section 5. Our system uses a MySQL database to implement the traffic archive (refer Figure 1) where traffic records are indexed by the flow end-time and the destination host of the flow. The system also maintains a MySQL database of host-level anomalies, indexed by the anomalous host IP and the timestamp of the anomaly. The prototype implements the immediate lookback option. Incubation parameter inference (Section 5.3) and iterative feature extraction (Section 5.4.2) are implemented as off-line processing capabilities.

We evaluated our prototype by emulating real-time traffic feeds collected over a 25-day dataset collected in Feb 2005 in the university network with over 16000 active hosts. The primary goal of this study is to confirm the practical viability of implementing and operating a system for uncovering causal attack flows based on our hypothesis. The detection, database, and correlation operations all run on a single dual Intel Xeon 2.8 machine, with 2 GB of RAM. The traffic records and the databases are stored on an external USB hard-drive. The memory requirements of our system are small, rarely exceeding 30 MB of RAM usage. We implemented two minor optimizations to minimize the database overhead. If there was a recent alarm on a host (i.e., within the last hour) and we have already performed the correlation step we avoid replicating the effort for the current alarm. To avoid insertions and retrievals to and from large databases, we create one database per hour of traffic to reduce the flow insertion and query time. Even without significant code optimizations, we find that we get 8X speed up (on average) over real-time traffic rates (it takes around 3 days to process the 25-day trace).

Processing overheads	
Total number of flows	616 million
Total time to process	$\approx 76hrs$
Avg. time to process one hour traffic	7.6 min
Max. time to process one hour traffic	14 min
Avg. host-based anomalies per hour	≈ 2600
Avg. queries to correlation engine per hour	45
Avg. incoming flows returned per lookup	116
Max. incoming flows returned per lookup	3795
Analyzing suspicious flows returned	
Number of suspicious flows identified	23663
Contribution of top-10 sources	72%
Contribution of top-10 destination ports	66%

Table 1: Results from a preliminary measurement study using our prototype using a 25-day trace

We present some measurements from the 25-day study in Table 1. On average, we find that it takes roughly 7.6 minutes to process a one hour traffic snapshot. However, at peak traffic rates it takes up to 14 minutes to process an hour's data. Even at peak load, the performance of our prototype is sufficient to keep up with observed traffic rates in a large university setting, which is comparable to large enterprise networks. The number of insertions from the anomaly detection system appears to be quite large relative to the total traffic volume. This is an artifact of the nature of the multi-resolution detection system. The detector reports an alarm for each constituent time resolution over sliding windows. Thus, it is often the case that the same host was reported as being anomalous on multiple resolutions at the same time. Also, a single conceptual anomaly (e.g. a host being anomalous over a 10 minute period) may appear as multiple anomalies during the interval of the anomaly (i.e., for each 10 second window within that 10 minute interval there will be an anomaly reported). The actual number of queries that were sent to the correlation step are much less than the number of raw anomalies reported, since we only need to process the first anomaly for that host within that anomalous interval. One surprising statistic is the maximum and average number of incoming flows returned for each such host-based anomaly. It turns out the majority of hosts flagged as anomalies were in fact server-type hosts as opposed to client-like hosts, and thus the number of incoming flows is orders of magnitude larger than those observed for normal client-like hosts (Section 4.1).

Over the 25 day period there were around 23000 suspicious (i.e., potentially causal) flows reported, which represents roughly 0.004% of the total number of flows. Analyzing these flows reveals that there are a small number of sources and application services that are dominant. For example, we notice that the top-10 application destination ports contribute roughly 66% of the reported flows. These included flows on ports 904 and 907 (unknown, 26%), 123 (NTP, 13%), 137-138 (NetBIOS, 9%), 135 and 445 (Windows RPC service known to be associated with scan traffic, 8%), and 7000-7001 (AFS, 7%). We observed that almost all the suspicious flows on ports 904 and 907 were from just three hosts in the network. These hosts appear to have been performing a linear scan of the address space during this period. Similarly, the top-10 sources alone contribute around 72% of the suspicious flows. These sources included four scanners (three scanning ports 904 and 907, one scanning ports 445 and 135), three NTP servers, two DNS servers, and one NETBIOS server. In addition to serving to confirm the operational feasibility of our approach, our study also provides some insights into the nature of events that may be identified when such a system is deployed.

8 Conclusions

We explored the hypothesis that combining host-based anomaly detection and flow-level correlation is a promising approach for determining attack causality. This hypothesis is based on the insight that inter-host communication patterns are sparse. Our approach relies only on coarse-grained network-level monitoring, without relying on prior knowledge of attack-specific properties and signatures. It is thus robust across a wide spectrum of worm attacks. We examined different aspects of the design space such as the different lookback strategies, the use of whitelists, automatic attack-feature extraction, and incubation inference. Our analytical study, trace-driven evaluations, and prototype implementation show that our approach provides the basis of a practical scheme for determining attack causality.

We conclude by pointing out limitations of our approach and summarizing some lessons from our evaluations.

Limitations: Attacks can obfuscate the notion of causality, by either employing a coordinated infection attempt (i.e., a host is infected by a combination of multiple incoming traffic flows), or by piggybacking malicious traffic on legitimate connections (e.g., malicious objects embedded in application data). Our approach does not address these attacks, but such attacks are also more difficult to implement and less potent in practice. Attackers can also introduce background traffic activity unrelated to the actual attack to mislead the correlation step, or alternatively launch multiple worms at the same time to achieve the same result. To deal with stealthy attacks, we introduced the incubation inference procedure. This process has limitations, especially when attacks are non-homogeneous (i.e., the scan-rate and incubation period vary across infected nodes). While our understanding of the limitations and attack-evasion strategies is not comprehensive, addressing these concerns provide interesting avenues for future work.
Lessons: First, the accuracy of host-based anomaly detection systems along the time-dimension i.e., providing an estimated infection time is an interesting component that has not been studied extensively in the literature. Second, while it is evident that having an understanding of attack parameters is useful, we demonstrate that automatically inferring information such as the port and incubation parameter without relying on prior knowledge of attack-specific information is feasible, and this inference can improve the accuracy. Third, understanding server and background scanning patterns within the network can significantly improve the accuracy by eliminating known non-causal events. Finally, we find that such whitelist information can reduce the dependence on accurate timing information and incubation inference. Specifically, we find that an infinite lookback approach when enhanced with the whitelist information can provide comparable accuracy to the immediate lookback approach (with an optimal lookback window). This suggests that incomplete information along one of the dimensions (timing, attack features, or background traffic knowledge) can be compensated by information along other dimensions.

References

[1] S. Staniford-Chen and S. Cheung and R. Crawford and M. Dilger and J. Frank and J. Hoagland and K. Levitt and C. Wee and R. Yip and D. Zerkle. Grids a graph-based intrusion detection system for large networks. In *19th National Information Systems Security Conference*, 1996.

[2] Argus. `http://www.qosient.com/argus`.

[3] Z. Chen, L. Gao, and K. Kwiat. Modeling the Spread of Active Worms. In *Proc. of IEEE INFOCOM*, 2003.

[4] Daniel R. Ellis, John G. Aiken, Adam M. McLeod, David R. Keppler, and Paul G. Amman. Graph-based Worm Detection On Operational Enterprise Networks . Technical report, MITRE Corporation, April 2006.

[5] Jayanth K. Kannan, Jaeyeon Jung, Vern Paxson, and Can Emre Koksal. Detecting hidden causality in network connections. Technical Report UCB/EECS-2005-30, EECS Department, University of California, Berkeley, 2005.

[6] H. Kim and B. Karp. Autograph: Toward Automated, Distributed Worm Signature Detection. In *Proc. of USENIX Security Symposium*, 2004.

[7] S. T. King, Z. M. Mao, D. G. Lucchetti, and P. M. Chen. Enriching Intrusion Alerts Through Multi-Host Causality . In *Proc. of NDSS*, 2005.

[8] A. Kumar, V. Paxson, and N. Weaver. Exploiting Underlying Structure for Detailed Reconstruction of an Internet-scale Event. In *Proc. of IMC*, 2005.

[9] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System . *Communications of the ACM*, 21:558–565, July 1978.

[10] P. McDaniel, S. Sen, O. Spatscheck, J. van der Merwe, W. Aiello, and C. Kalmanek. Enterprise Security: A Community of Interest Based Approach. In *Proc. of NDSS*, 2006.

[11] John McHugh and Carrie Gates. Locality: A New Paradigm for Thinking About Normal Behavior and Outsider Threat. In *Proc. of the Workshop on New Security Paradigms*, 2003.

[12] J. Newsome and D. Song. Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software. In *Proc. of NDSS*, 2005.

[13] Ruoming Pang, Mark Allman, Mike Bennett, Jason Lee, Vern Paxson, and Brian Tierney. A First Look at Modern Enterprise Traffic . In *Proc. of IMC*, 2005.

[14] M. A. Rajab, F. Monrose, and A. Terzis. Worm Evolution Tracking via Timing Analysis. In *Proc. of ACM CCS WORM*, 2005.

[15] M. Roesch. Snort - Lightweight Intrusion Detection for Networks. In *Proc. of USENIX LISA*, 1999.

[16] Stuart E. Schechter, Jaeyeon Jung, and Arthur W. Berger. Fast Detection of Scanning Worm Infections. In *Proc. of RAID*, 2004.

[17] Vyas Sekar, Yinglian Xie, Michael K. Reiter, and Hui Zhang. A Multi-resolution Approach for Worm Detection and Containment. In *Proc. of IEEE/IFIP DSN*, 2006.

[18] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the Internet in Your Spare Time. In *Proc. of USENIX Security Symposium*, 2002.

[19] S. Staniford-Chen and L. T. Heberlein. Holding Intruders Accountable on the Internet. In *Proc. of the IEEE Symposium on Security and Privacy*, 1995.

[20] T. Toth and C. Kruegel. Connection-history based anomaly detection. In *Proc. of the IEEE Workshop on Information Assurance and Security*, 2002.

[21] X. Wang, Z. Li, J. Xu, M. K. Reiter, C. Kil, and J. Y. Choi. Packet vaccine: Black-box exploit detection and signature generation. In *Proc. of ACM CCS*, 2006.

[22] D. Whyte, E. Kranakis, and P. C. van Oorschot. DNS-based Detection of Scanning Worms in an Enterprise Network . In *Proc. of NDSS*, 2005.

[23] Matthew M. Williamson. Design, Implementation and Test of an Email Virus Throttle. In *Proc. of ACSAC*, 2003.

[24] J. Wu, S. Vangala, L. Gao, and K. Kwiat. An Effective Architecture and Algorithm for Detecting Worms with Various Scan Techniques. In *Proc. of NDSS*, 2004.

[25] Y. Xie, V. Sekar, D. A. Maltz, M. K. Reiter, and H. Zhang. Worm Origin Identification Using Random Moonwalks. In *Proc. of IEEE Symposium on Security and Privacy*, 2005.

[26] Vinod Yegneswaran, Paul Barford, and Johannes Ullrich. Internet Intrusions: Global Characteristics and Prevalence. In *Proc. of ACM SIGMETRICS*, 2003.

[27] Y. Zhang and V. Paxson. Detecting Stepping Stones. In *Proc. of USENIX Security Symposium*, 2001.