Forensic Analysis for Epidemic Attacks in Federated Networks

Yinglian Xie Vyas Sekar Michael K. Reiter Hui Zhang

August 31, 2006
CMU-CyLab-06-014

CyLab
Carnegie Mellon University
Pittsburgh, PA 15213

Forensic Analysis for Epidemic Attacks in Federated Networks

Yinglian Xie Vyas Sekar Michael K. Reiter Hui Zhang
Carnegie Mellon University
{ylxie, vyass, reiter, hzhang}@cs.cmu.edu

September 5, 2006

Abstract

We present the design of a Network Forensic Alliance (NFA), to allow multiple administrative domains (ADs) to jointly locate the origin of epidemic spreading attacks. ADs in the NFA collaborate in a distributed protocol for post-mortem analysis of worm-like attacks. Information exchange between any two participating ADs is limited to traffic records that are known to both sides, maintaining the privacy of participants. Such an architecture is incentive-compatible – participants benefit by gaining better local investigative capabilities, even with partial deployment. Further, we show that by sharing local investigation results, ADs can achieve global investigative capabilities that are comparable to a centralized implementation with access to global traffic records. Our evaluation demonstrates that it is feasible for large-scale attack investigation to be incrementally deployed in an Internet-like federation.

1 Introduction

Attackers today can launch epidemic attacks (worms) of dramatic intensity and reach. Much effort has been spent on designing more effective detection mechanisms (e.g., [1]), building better defense measures (e.g., [2]), and generating vulnerability-specific remedies (e.g., [3]). While these approaches provide immediate benefit to operators and end users, there is little ability to perform post-mortem diagnostic and forensic investigation of these incidents. Specifically, a service provider or enterprise needs a means to determine the entry point of a worm to its network so that the vulnerability it initially exploited can be identified. In addition, the ability to determine the true origin of a worm on a large network such as the Internet could help to prosecute and, ultimately, deter these attacks.

In this paper, we address the problem of providing forensic capabilities for investigating large-scale attacks in distributed, federated networks. A federated network is composed of multiple independent Administrative Domains (ADs). Conceptually, an AD is a network under a single administrative authority, and a single AD may be composed of one or more Autonomous Systems (ASes) in a BGP context. The most popular instance of a federated network is the Internet as we know it today, which is composed of multiple (possibly competing) ISPs, with diverse economic and peering relationships.

We propose the concept of a Network Forensic Alliance (NFA) [1]—a collaborative effort involving multiple ADs to provide network-wide and localized forensic capabilities, as depicted in Figure 1. Using the techniques we describe in this paper, an NFA allows multiple ADs in a federated network to jointly perform local attack investigation, with the additional ability to diagnose attacks globally.

Each participating AD in the NFA possesses an independent traffic monitoring infrastructure. Traffic data (in the form of flow records) are collected and saved for a period of time during which an investigation can be launched using them. Investigative capabilities are realized using a distributed protocol which builds on, and enhances our earlier work [5] on worm origin identification. To launch an investigation, each AD feeds its local traffic records to its analysis engine that runs our protocol (described in Section 4). In the course of running the

[1] The name is inspired by existing efforts at a Fingerprint Sharing Alliance [4], for generating and sharing attack signatures.

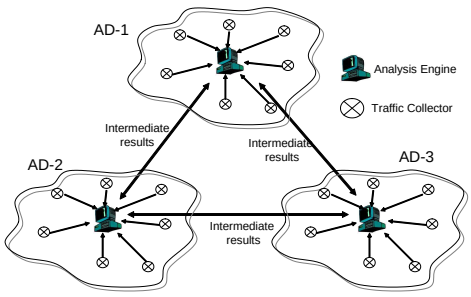


Figure 1: NFA: Architecture for federated forensics

protocol, the analysis engine may communicate intermediate results to other participating ADs. Each analysis engine subsequently post-processes its results to obtain a local view of the attack entry within its domain, or shares the results with other domains to obtain a global view of the attack.

To our knowledge, we are the first to formulate the problem of forensic analysis for epidemic attacks in the context of federated networks. Our design is not only attack-agnostic (within the class of epidemic-spreading attacks), but also addresses deployment concerns such as privacy, participation incentives, and incremental deployment. The NFA: (i) is privacy-preserving, in that each AD reveals to another AD only records for traffic that each has already seen; (ii) provides participation incentives for individual ADs to collaborate with other ADs, i.e., each AD improves its own local diagnosis by cooperating with others; and (iii) offers strong incremental deployment properties, in that participants benefit even in partial-deployment scenarios. We consider (i)-(iii) to be essential to a viable federated approach, given the reticence of competing ADs to cooperate except when it is in their interests to do so.

We also present a methodology to address the fundamental challenges of evaluating the effectiveness of worm forensics in such a distributed setting, given the non-availability of multi-AD traffic datasets. Our framework permits us to specify well-defined measures of incentives that any solution in this domain will need to address. Our evaluation includes two sets of experiments: one uses simulation on a network topology defined by the Internet AS-level topology, and the other uses traffic collected from the *Internet2* [6] educational backbone. These experiments show that our approach realizes investigative capabilities comparable to an ideal unified network administrative model, while operating in a federated setting. The evaluations also validate that our approach greatly increases the local investigative capabilities of participating ADs; enables ADs to generate a network-wide reconstruction of the epidemic attack; and degrades gracefully with only partial deployment.

2 Background

Early work in network forensics focused on stepping-stone detection and IP traceback. Stepping-stone detection attempts to bridge sources of attack indirection, in attacks that are launched through a series of intermediate machines. Existing techniques (e.g., [7, 8]) typically require packet-level analysis, and in some cases require the analysis to be applied close to the compromised machines. Both of these would be obstacles in applying these techniques to trace the origin of a large-scale worm after the fact, and our approach requires neither. IP traceback focuses on identifying the true sources of packets with spoofed source addresses (e.g., [9, 10, 11]). With respect to large-scale epidemic attacks, traceback techniques typically are not needed since source addresses are seldom spoofed. However, even in epidemic attacks with spoofing, these techniques would simply identify another victim in the vast majority of cases. It is this last obstacle that we seek to overcome here.

Network telescopes have been suggested as an alternative approach to reconstruct worm attacks [12, 13]. Kumar et al. [12] present an approach for reconstructing the spread of the Witty worm by using scan traffic collected by network telescopes to reverse engineer the pseudorandom number sequences used. Rajab et al. [13] use the telescope-observed scan traffic to infer infection times by studying the inter-arrival times between successful scans. Both approaches tackle the problem of worm forensics, but are applicable to only

random-scanning attacks that generate traffic to network telescopes. Furthermore, the robustness of using network telescopes for forensic analysis is a topic of ongoing research [14].

Our earlier work proposed a random moonwalk algorithm for identifying the origin of an epidemic attack [5]. The algorithm takes traffic flow-records as inputs, and generates as output a set of "high-ranking" flows. The algorithm works by repeatedly performing random "moonwalks" on the inter-host communication graph and correlating the set of flows traversed by the walks. Each moonwalk starts from an arbitrarily chosen flow. The algorithm then randomly picks a next step backward in time from the set of flows that arrived at the source host of the current flow within the previous Δt seconds. For each next step, the above process repeats until there are no candidate flows to continue the path, or the walk has traversed a maximum of d hops. Our analysis and empirical evaluation [5] suggest techniques for selecting the sampling window size Δt and the maximum path length d, using a trace-driven adaptive approach. After many such walks, the algorithm returns a set of flows that are most frequently traversed by the walks. Due to the tree-structured nature of epidemic attacks, the initial causal flows of the attack emerge among the highly ranked flows returned.

The random moonwalk algorithm assumes a unified network administrative model. In large federations such as the Internet, establishing a centralized traffic repository is hardly feasible due to privacy and economic considerations of multiple competing ADs. The approach we propose here builds from the random moonwalk algorithm, but overcomes the shortcomings for deploying this algorithm in Internet-like federated environments. While each AD can perform *isolated* attack investigation using the centralized algorithm, this does not achieve global forensics goals. In addition, as we will show, isolated forensics provides weaker local investigation capabilities than the distributed protocol we propose here, since each AD observes only a small portion of a large-scale, distributed attack.

3 System Design

In this section, we motivate the key components of our design, and outline the three stages that comprise the distributed operation of an NFA.

3.1 Why Random Moonwalks

The key insight behind our design decision of adopting the random moonwalk algorithm is an alternative interpretation of it as a Monte-Carlo simulation method to compute the largest eigenvector of a network flow graph. In this light, known results suggest that this algorithm should be relatively resilient to missing data, and so could form the basis of a solution offering good partial deployment properties.

To make this alternative interpretation more concrete, consider a directed *flow-graph* G_f, where each node represents a flow. Given a sampling window size Δt for random moonwalks, we insert a directed edge from node n_2 to node n_1, if the flow n_1 arrived at the source of the flow n_2 at most Δt time units before the start of the flow n_2. That is, there is a directed edge from node n_2 to node n_1 if a random moonwalk could (single-) step from the flow n_2 to the flow n_1. After inserting all such edges, we then insert an edge from each sink n_2 (i.e., with zero out-degree) to every other node in G_f. A random moonwalk then corresponds to a Markov random walk on G_f, where at each node n_2, the distribution of the next node is uniform over all nodes to which n_2 has an edge. With this interpretation, the normalized adjacency matrix A_f defined by G_f is a stochastic matrix whose row sums are all $\mathbf{1}$. We consider the spectral analysis of matrix A_f, namely (non-trivial) solutions to the equation:

$$A_f x = \lambda x$$

The stationary distribution π_0 of A_f, computed as the largest eigenvector of A_f, represents the probability of reaching every node in a global stable state. The flows with the largest probabilities of being traversed in a random moonwalk, thus correspond to those nodes in G_f which have the largest numerical values in the largest eigenvector (i.e., the stationary probability distribution). These flows, with high probability, are the initial causal flows. We can therefore view random moonwalks as a Monte Carlo sampling method to compute the largest eigenvector of the flow graph G_f.

Such a spectral analysis view suggests that random moonwalks can be robust to missing traffic, as previous studies have shown that spectral techniques are effective in identifying the underlying structure of graphs even with partial data availability [15, 16]. For this reason, we choose to build the NFA from this random moonwalk algorithm, extending it into a distributed version that can operate in a federated environment.

3.2 Federated Forensics

In the NFA, multiple ADs independently collect traffic data, and jointly perform attack investigation to identify both the local attack entry points and the global attack source. The high-level idea is that multiple domains can perform loosely coordinated random moonwalks inside their own domains, so that when these walks are viewed together, they achieve the same effect as performing random moonwalks on a unified network. This process involves the following three stages:

1. Distributed traffic monitoring: Each participating AD logs flow-level traffic records. Prior work on hash-based traceback has demonstrated that it is feasible to build such fine-grained traffic logging capabilities [11]. As shown in Figure 1, each AD deploys traffic collectors (e.g., routers) to log network flow records, which will then be accessed or queried by an analysis engine.

2. Collaborative random moonwalks: Federated domains perform joint attack investigation via a distributed protocol. The analysis engines interact with each other to launch coordinated local random moonwalks on the traffic collected within their own domains. Intermediate results will be shared among ADs to guide new local random moonwalks in an iterative fashion. These intermediate results should not release data that are proprietary to an AD or that should otherwise be protected.

3. Post-processing for attacker identification: After performing the collaborative random moonwalks, each participating domain obtains a set of suspicious flows. Each participant can either choose to use the flows independently for local attack investigation, or to further share information to achieve global investigative capabilities.

Since the NFA would be deployed incrementally, one challenge throughout the above process is to gracefully handle missing information, so that the NFA utilizes all available traffic records even when certain ADs do not participate. Next, we first describe how multiple ADs jointly perform collaborative random moonwalks and perform post-processing. We then discuss strategies to deal with missing traffic records in partial deployment scenarios.

4 Protocol for Distributed Random Moonwalks

In this section, we present a distributed protocol for multiple domains to jointly perform attack investigation in a federated network. In the distributed protocol, each AD participating in the NFA performs two operations: (1) it runs local random moonwalks on flow-records it has collected, and (2) it exchanges intermediate results with other collaborating ADs. These intermediate results will ensure that moonwalks continue even when they cross the network boundaries of ADs which originated them. When these local moonwalks are "stitched" together (via the inter-AD exchanges), they approximate random moonwalks on a global traffic trace (i.e., the union of the traffic observed at all the participating ADs). Throughout the protocol, each AD independently keeps a count of the number of times each flow has been traversed in the local random moonwalks. When the protocol terminates, each participating domain will be able to identify a set of Z top frequency flows among their traffic records as candidate flows for further investigation. To simplify the description of the protocol, we assume that all ADs in the network collaborate in the distributed operation, and defer a discussion of partial deployment until Section 6.

The protocol begins with a relatively lightweight bootstrap phase to initialize parameters. Participating ADs first decide on the time boundary for investigating the attack. Next, they select `H`, the maximum number of AD-hops each distributed moonwalk can traverse, and the sampling window size Δt. Δt specifies the maximum look-back window within which candidate flows can be selected for continuing moonwalks. Each domain (AD_i) could also independently specify a maximum local walk-length `d(i)` for walks it performs inside its

Notation	Description
`Flow(i,j)`	A flow record with the source-host in domain AD_i and the destination-host in domain AD_j.
`FlowSet(i)`	The set of network flow records whose sources belong to AD_i
`InitSet(i)`	A multiset of flow records and their AD hop counts that AD_i uses to start local moonwalks.
`W(i)`	The number of walks initially started by AD_i.
`d(i)`	The maximum length of a local random moonwalk within AD_i.
`H`	The maximum number of AD hops a distributed moonwalk can traverse.
Δt	The sampling window size, defined as the maximum time window to look back for continuing a moonwalk.

Table 1: Notation used in the description of the distributed protocol. `W(i)` and `d(i)` are local parameters defined by AD_i. `H` and Δt are global parameters participating ADs decide before the start of the protocol.

own domain. ADs also agree on a traffic normalization factor, η, which determines the number of moonwalks that each AD launches initially.

After the initialization, each AD_i participates in the distributed random moonwalk protocol, and its operation in the protocol is as defined by Figure 2. Using the notation in Table 1, the protocol works as follows:

1. Initial launch: Each AD_i initially launches random moonwalks from within its domain. The number of walks `W(i)` launched by AD_i, is calculated as `W(i)` = $\eta\times$ |`FlowSet(i)`|. AD_i then selects `W(i)` flows from `FlowSet(i)`, uniformly at random with replacement. For each selected flow `F`, AD_i sets its AD hop count `h`$= 0$, constructs the tuple `<F,h>`, and inserts it into `InitSet(i)` [2].

In our two-domain example (Figure 3), both AD_1 and AD_2 use `W(i) = 2`. During this initial step, AD_1 selects flow `A(1,1)` twice and inserts two entries into `InitSet(1)`, both with hop count set to zero. AD_2 similarly inserts two entries into `InitSet(2)`.

2. Local random moonwalks: For every entry `<F,h>` in `InitSet(i)`, AD_i will launch a random moonwalk, with `F` as the initial step. After performing the local moonwalk from `F`, AD_i deletes the corresponding tuple `<F,h>` from `InitSet(i)`. A walk will stop within domain AD_i if it meets any of the following criteria.

- There is no flow to continue the walk in the previous Δt seconds.
- The walk has traversed the maximum number of local-hops `d(i)` within AD_i.
- The walk has reached a cross-domain flow whose source host is located in a different AD.

The first two stopping criteria are the same as in the centralized algorithm [5]. The third condition arises due to the distributed setting. Since each AD has only a local view of the global traffic, it lacks sufficient information to select a next step to continue the moonwalk. Specifically, for flows originating from foreign hosts that belong to other ADs, AD_i cannot observe all possible incoming flows at such hosts. In the example, a random moonwalk starting from flow `A(1,1)` stopped within AD_1 because it reached flow `C(2,1)`, whose source is inside AD_2. A moonwalk starting from flow `A(2,2)` stopped within AD_2 after several steps because there was no flow to continue the walk within the previous Δt seconds.

3. Cross-domain exchanges: If a local random moonwalk starting from `F` reaches a cross-domain flow `Flow(j,i)` [3], AD_i increases the corresponding AD hop count `h` by one. If the new `h` is smaller than the maximum AD hop count `H`, AD_i sends the tuple `<Flow(j,i),h>` to AD_j.

On reception of the tuple `<Flow(j,i),h>`, AD_j inserts it into `InitSet(j)`. AD_j will subsequently launch a local random moonwalk starting from `Flow(j,i)`, to continue the distributed random moonwalk.

For example, AD_1 sends `<C(2,1),1>` to AD_2, which then inserts the tuple into `InitSet(2)`. AD_2 does not send any flow record to AD_1 because the local moonwalk starting from flow `A(2,2)` does not reach a cross domain flow.

[2] We use multisets for keeping tuples of flow records and their AD hop counts. Each tuple can occur multiple times in a multiset.

[3] We assume that the source-AD information for each flow can be obtained from `whois` lookups, data from public routeservers, or using existing tools [17].

```
// delta_t denotes the sampling window size,
// H is the maximum AD hop count
// TSet is a data-structure implementing a set
TSet distributed_moonwalk(TSet FlowSet(i), int W(i),
                          int d(i),int H,int delta_t){
  // Initialize InitSet(i)
  InitSet(i) = {};

  // Select W(i) flows to insert into InitSet(i)
  for (int j = 1 to W(i)){
    // Select a flow F randomly from FlowSet(i)
    F = get_random_flow(FlowSet(i));
    InitSet(i) = InitSet(i) U {<F,0>};
  }

  // Perform local moonwalks
  while (InitSet(i) not empty){
    foreach tuple <F,h> in InitSet(i) {
      // Perform a local random moonwalk starting at F
      // Let endflow be the last step of a local walk
      endflow = random_moonwalk(F, d(i), delta_t);

      // Remove <F,h> from InitSet(i)
      InitSet(i) = InitSet(i) - {<F,h>};

      // Check if the walk reached the max. AD hopcount
      h = h + 1;
      if (h >= H)
         continue;

      // If the source host of endflow is in AD(j),
      // Send the tuple <endflow,h> to AD(j) so that
      // AD(j) will continue the walk from endflow
      int j = find_source_domain(endflow);
      if (j != i){
        send <endflow,h> to AD(j);
      }
    } // end foreach
  } // end while
  Return a set of flows with highest traversal counts;
}

// tuple <F,h> is the message received from another AD
void received_flow_record(Tuple <F,h>){
  // Insert <F,h> into InitSet(i)
  InitSet(i) = InitSet(i) U {<F,h>}
}
```

Figure 2: Pseudocode for the operations each AD_i executes in the protocol. Each analysis engine performs two operations of running distributed random moonwalks and updating its InitSet(i).

4: Termination condition: If `InitSet(i)` becomes empty, AD_i terminates its operation, and the analysis engine can then identify the set of Z flows with the highest counts (i.e., the sum of counts across all of its local random moonwalks).

The inter-AD exchanges have two important properties: (1) minimal information disclosure, and (2) low overhead. The flow records that can be exchanged between a pair of ADs can only be among the set of inter-domain flows between them. Since these cross-domain flows will be independently logged by both ADs, exchanging these flow records does not reveal internal traffi c information that is not already available to either AD.

The worst-case communication overhead occurs when every step along the walk triggers a message to be sent across a pair of ADs, resulting $\sum_{i=1}^{n}\mathtt{W(i)} \times \mathtt{H}$ flow records to be exchanged in the federation. $\sum_{i=1}^{n}\mathtt{W(i)}$

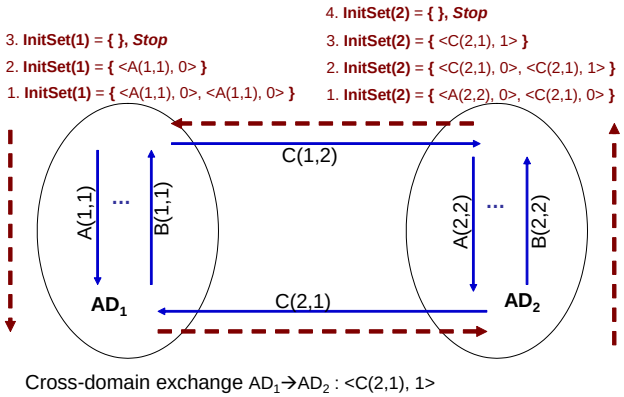


Figure 3: A simple example with two domains collaborating in the distributed protocol. Solid arrows represent the direction of network flows, and dashed arrows represent the direction of random moonwalks (reversed with respect to the flow directionality).

will typically be a small fraction of traffic, depending on the traffic normalization factor η (typically of the order of 0.1-1%). So the number of records that need to be exchanged will be a very small fraction of total unique flow records. However each flow-record that needs to be exchanged incurs a per-record communication overhead between the pair of ADs. To reduce the number of communication events, ADs can choose to aggregate the set of cross-domain flows at which local walks terminate, and send these flow-records in a single communication exchange with another collaborating AD.

5 Post-Processing for Investigation

After performing the distributed random moonwalks, each participating domain obtains a set of top frequency flows, to serve as starting points of further attack investigation. Participating ADs can either choose to use these flows for independent local attack investigation, or further share information to achieve global investigative capabilities.

5.1 Local View

ADs can use the flow counts obtained after the distributed moonwalks on the set of flows collected within their network boundaries, and compute the set of Z top frequency flows for local diagnosis. The locally observed top frequency flows can then be used to diagnose the initial attack entry points, or initial infected hosts within a domain. Identifying the attack entry points can help detect configuration or firewall rule errors that may have allowed the attack to enter the network perimeter in the first place.

With such a *Local View* option, ADs realize benefits of collaboration by just participating in the distributed protocol. No additional information will be shared after the distributed protocol has been executed. Section 7.1.2 quantitatively evaluates the participation benefit in terms of improved quality of local attack investigation compared with isolated forensics using various measures.

5.2 Global Merge

ADs can choose to share suspicious flow records identified within their domains, for the NFA as a whole to realize global benefits. This functionality is comparable to operation under a unified network model to reconstruct the initial stages of the attack, providing diagnostic aids to pinpoint the origin of the attack.

In such a *Global Merge* strategy, each AD_i identifies Z top frequency flows from `FlowSet(i)`, and broadcasts the counts of these locally identified flows to every other participating domain, after executing the distributed protocol. After receiving the top frequency flows from all participating domains, each AD can identify the Z flows with the highest global counts.

We note that to compute global flow ranks, each AD_i contributes Z flows from `FlowSet(i)`, instead of

all collected flows that are used in Local View. Since for $i \neq j$, `FlowSet(i)` and `FlowSet(j)` do not overlap, flows will not be double counted by different domains. For each flow in `FlowSet(i)`, AD_i is the only domain to select its next step in the distributed random moonwalks; thus the frequency count that AD_i maintains for that flow is the flow's global frequency count. Therefore, the computed global flow ranks will be equivalent to the ranks computed by an ideal centralized moonwalk algorithm, assuming the entire federated network is a large unified network.

ADs need not reveal internal traffic records that have been identified, and can share only high frequency cross-domain flow records by broadcasting these flow records and their counts to other participants. This can provide sufficient information to understand how the attack propagated across domains. Participating ADs can also choose to reveal these selected flow records using different degrees of source-address anonymization. The extent to which the globally reconstructed initial attack graph aids further diagnosis, depends on the degree of anonymization that the participating ADs engage in. At one end of the spectrum, ADs can choose to construct the graph at the granularity of actual end-hosts, providing the most fine-grained diagnosis possible. At the other end, ADs can choose to construct the graph at the AD granularity. In the intermediate space, ADs can reconstruct the attack graph at the granularity of coarse-grained network prefix ranges (e.g., using /8 or /16 masks).

6 Handling Partial Deployment

So far, we assumed that all the ADs in a federated network participate in the NFA. In reality, such an NFA will be deployed in an incremental fashion. Even though the NFA may continue to observe a significant fraction of traffic, we need to extend the protocol to utilize all available traffic records in partial deployment scenarios. Specifically, during the distributed random moonwalks, it may not be possible to continue the walk from flows whose source-ADs do not participate in the NFA.

6.1 Impact on Attack Coverage

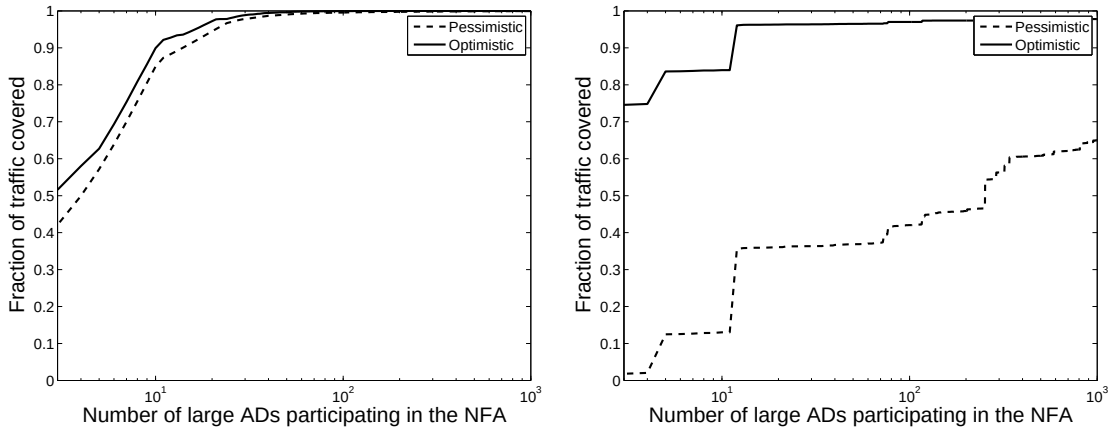
The first question is coverage over attack traffic. If the NFA is incapable of observing some or most of the attack traffic, then most of the post-mortem analysis will be rendered ineffective. To gain some preliminary insight on the impact of partial deployment, we use data from Routeviews [18] to analyze the amount of traffic that will be available to ADs in the NFA under partial deployment.

First, we obtain the AS-level topology by inferring inter-AS links from the AS-path attributes of BGP announcements. Next, we obtain for each valid AS number, the advertised address ranges observed at the Routeviews route-server. Because route-advertisements may have overlapping address ranges [4], we process it to assign only unique non-overlapping ranges to each AS. For our analysis, we assume that the traffic between ASes follows a gravity model, i.e., the traffic between two ASes A and B, is proportional to $|A| \times |B|$, where $|A|, |B|$ represent the size of ASes in terms of address space ownership. Note that for scanning worms, the gravity model closely resembles the actual attack traffic distribution. Note that even if two ADs A and B are both non-participants, other participants which transit traffic on the path between A and B may still be able to observe the inter-domain traffic between A and B. In the following analysis, we assume shortest-path routing over the AS-level topology

In Figure 4, we depict two partial deployment scenarios, and assume that the top-k largest ADs participate in the NFA. We consider both a pessimistic estimate and an optimistic estimate. The pessimistic estimate is obtained by assuming both intra-AD traffic within non-participating ADs and all inter-domain traffic between non-participating ADs is unavailable to ADs the NFA. The optimistic estimate is obtained by using routing information to find out if any of the participating AD is along the shortest-path between non-participating ADs, and thus can audit their inter-domain traffic.

When the top-k ADs with largest address-space ownership participate, we find that the network traffic covered by these ADs constitutes a significant fraction of the overall traffic. In this case, the optimistic estimate

[4] For traffic engineering, an AS may announce both a specific prefix range and a larger prefix range that includes the specific range



(a) By address-space

(b) By topological degree

Figure 4: Traffic coverage under different partial deployment scenarios

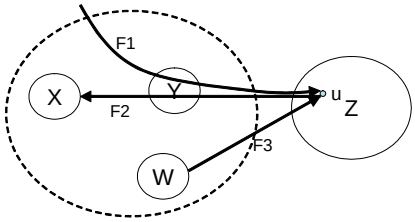


Figure 5: An example partial deployment scenario. W, X, Y are three ADs that are in the federation, and AD Z does not participate. F1, F2, and F3 represent three network flows to or from host u in AD Z.

is not very different from the pessimistic estimate, since the ADs with large address-space ownership contribute a large volume of traffic under the gravity model. With the degree-based participation, the optimistic estimate is significantly better than the pessimistic estimate. This implies that the transit ADs can help the NFA obtain significant coverage of traffic between non-participating ADs also.

6.2 Recovery Strategies

Even though the NFA may continue to observe a significant fraction of traffic, it is not entirely obvious how the distributed random moonwalks can handle partial deployment. Specifically, during the distributed random moonwalks, it may not be possible to continue the walk from flows whose source-ADs do not participate in the NFA.

Figure 5 shows an example partial deployment scenario, where ADs W, X, Y participate in the federated network, while AD Z does not. In the distributed moonwalks, suppose X performs a local moonwalk that reaches flow F2, whose source host u is located inside domain Z, then none of the participating ADs will be able to observe all the incoming flows to u to continue this walk across domains.

A naive *discard* solution is to assume a closed world environment where ADs simply discard flows originating at non-participating domains. This solution is sub-optimal, as it does not utilize all the observed traffic data among all collaborating ADs, and interesting traffic flow records may end up being discarded in the process. A better solution is to try and perform a best-effort next step selection based on the data available to the participants in the federation. We consider the following strategies to enhance the distributed random moonwalks for continuing a walk once some participating AD reaches a flow originated from a non-participating domain.

Random selection: The simplest strategy is to randomly pick a participating AD and request it to continue the walk. In our example, X could randomly pick W or Y to continue the walk from F2, based on their local views of the incoming flows to host u.
Routing-path based selection: The second option for the AD is to use routing information and select the participating AD that is closest to the source host on the AD-path. The assumption here is that this selected AD will have the best available partial view to continue the walk. In our example scenario, flow F2 is routed through AD Y to X. Hence with routing-based selection, X would send a message to Y, requesting it to continue the walk, instead of choosing AD W.
Routing-horizon based selection: A generalization of the routing-based selection is to identify a set of participating ADs, defined as a *routing horizon*, that provide the greatest coverage over routing paths from the non-participating AD. Given the routing horizon, the AD that reaches a cross-domain flow in the random moonwalks can broadcast the flow record among the participating ADs on the routing horizon, for selecting candidate next step flows. Recall that given the current step f of the walk, candidate flows are those whose destination is the source of f, and which occur within previous Δt seconds of f. From the set of candidate flows, the AD can randomly select a flow to continue the walk, and hand off the current flow to the source AD of the selected flow. In our example, when AD X reaches flow F2, it could query both W and Y for their observed incoming flows to host u within the previous Δt seconds to F2. Suppose both F1 and F3 are such candidate flows, X could then randomly pick one, say F1 as the next moonwalk step.

7 Evaluation: Simulation Study

Our primary results are drawn from a simulation study, with synthetically generated traffic traces mapped into an Internet AS-level topology with over 16,000 ASes (Section 7). We also independently validate our findings using data derived from the *Internet2* educational backbone (Section 8). Broadly, our evaluation attempts to answer the following questions:

- How does the performance achieved in a federated setting compare with similar analysis in a unified network model? What is the overhead of message exchanges in the distributed protocol? (Section 7.1.1)
- Is the architecture incentive-compatible, i.e, what are the benefits an AD can gain by collaborating with other ADs? (Section 7.1.2)
- How does the performance degrade under different partial deployment scenarios? Is the architecture incrementally deployable, and in particular do participants realize benefits even under partial deployment? (Section 7.1.3)

7.1 Simulation Setup

We use a simulated trace-driven study to evaluate the deployability of an NFA in an Internet-scale large federation. The network topology for our study is defined by the Internet AS-level topology, obtained from RouteViews [18]. Each AS in the AS-level topology is mapped into a single AD in our federated network. We model the address-space ownership distribution across different ADs using prefix-ownership advertisements observed from RouteViews, proportionally scaling them down to a smaller address space of 10^7 hosts. Routing in our federated network is modeled using hop-count based shortest-path routing.

We implemented an event-driven simulator to generate synthetic flow-level traffic traces in our simulated network topology. Each of the 10^7 end-hosts in the address space is associated with a *working-set* of destination end-hosts that it talks to. The working-set size distribution across the hosts is captured using a discretized power-law distribution, with working-set sizes ranging between 10 and 1000 (the power-law ensures that most of the hosts have small working-sets and a small number of hosts have large working-sets). To initiate a flow, a host picks a destination from its designated working set using a preferential selection policy (also modeled with a discretized power-law distribution over the working-set size).

Attack traffic is specified using a worm scanning model, a scanning rate, and the fraction of hosts vulnerable. In our evaluation, vulnerable hosts are selected uniformly at random from the address-space of 10^7 hosts with

a total of 10% of hosts vulnerable to the attacks [5]. We use two different worm scanning models: random and local-preferential scanning. The random-scanning worm selects destinations uniformly at random from the entire address-space. The local-preferential scanning worm selects destinations to infect based on topological locality. For example, an infected host will try and infect hosts within the same source AD with a higher probability, and this probability decreases as a function of topological proximity (measured in terms of hop-count).

For each of the following experiments, we perform 5 independent runs and report the mean. The standard deviations observed across the different runs were small and are not reported for brevity. We set the traffic normalization factor η in the distributed random moonwalks to 10^{-3}. For each experiment, we use the methodology in [5] to select the optimal sampling window size, and set the maximum path length $d(i) = 20$ for each local moonwalk inside a domain.

7.1.1 Accuracy and Overhead

We first compare the performance of the distributed random moonwalks with a centralized approach, by varying the worm propagation rate with a random scanning attack. The centralized approach [5] assumes access to global traffic records from all ADs in the network. The collaborative, distributed protocol is as described in Section 4. The performance of the centralized and distributed approaches is measured in terms of the detection accuracy, defined as the fraction of the top Z returned flows that are actually causal flows. With the distributed algorithm, these Z flows are identified using the *Global Merge* procedure presented in Section 5.2.

Figure 6 shows that the detection performance achieved by the distributed random moonwalks closely approximates that of the centralized algorithm. The worm scanning rates are presented relative to the mean of the normal per-host traffic flow rates. Both the centralized algorithm and the collaborative approach achieve high detection accuracy regardless of the worm rates. For the rest of our evaluations, we only present results from a single worm rate, with a scanning rate equal to six times the mean normal traffic rate.

We show in Figure 7, the detection accuracy and communication overhead as a function of H, the maximum number of AD-hops that moonwalks may traverse. Figure 7 (a) shows how the performance approaches that achieved by the centralized algorithm, as H increases. With a small H, the distributed version is less accurate in identifying causal flows. The reason is that the initial steps of the moonwalks start at random flows, and non-causal attack flows are inherently more numerous than causal flows. As ADs exchange intermediate results over time, the majority of the returned flows are causal flows, and the detection accuracy converges after five or six hops. The communication overhead (Figure 7 (b)), in terms of the total number of flow records exchanged in the network, is on the order of 10^{-3} of the total traffic records in the network. While the overhead is a monotonically increasing function of the accuracy, it does not increase significantly beyond the convergence limit of five or six hops.

We further examine how the communication overhead distributes across the participating ADs in this case. Figure 8 (a) shows the fraction of the total exchanged flow records that each participating AD receives in the course of the distributed protocol. The distribution is highly skewed, with a small number of ADs receiving a large fraction of the total flow records exchanged in the network. Figure 8 (b) shows a similar trend in terms of the number of ADs that each participating AD needs to communicate with in the protocol. Further investigation shows that the ADs which receive a high volume of flow-records and need to communicate with a large number of collaborating ADs are ones that own large chunks of the overall address space (and also contain a large fraction of infected hosts). Thus a lot of flows traversed by random moonwalks originated from these domains.

7.1.2 Benefits of Participation

For a federated architecture to be viable, each participating AD must benefit from collaborating with other ADs. Here, we consider the case where ADs collaborate in the distributed random moonwalks, but subsequently choose to perform only the *Local View* (Section 5.1) for post-processing. This is compared against an isolated

[5]The random moonwalk algorithm is robust to the fraction of vulnerable hosts [5]. We do not duplicate the results here due to space constraints.

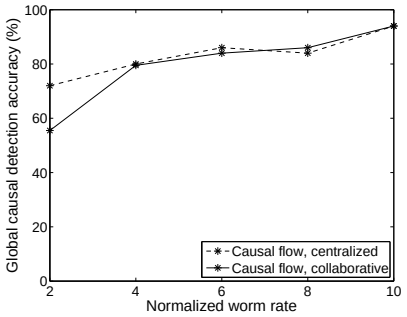
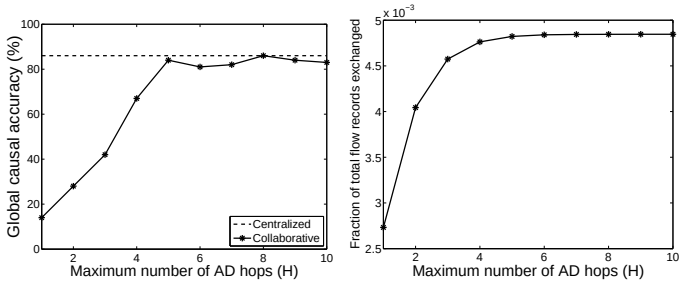
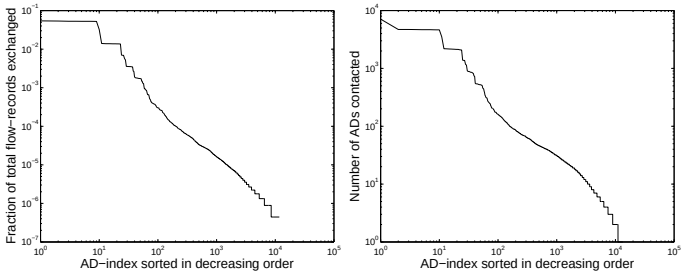


Figure 6: Global causal flow detection accuracy with the distributed protocol in the NFA compared with the centralized implementation in a unified network model. We vary the worm rate (relative to the mean normal traffic rate), and select the top $Z = 100$.



(a) Detection accuracy (b) Communication overhead

Figure 7: Accuracy and communication overhead increase as a function of the maximum AD-hops H a moonwalk can traverse, ($Z = 100$).

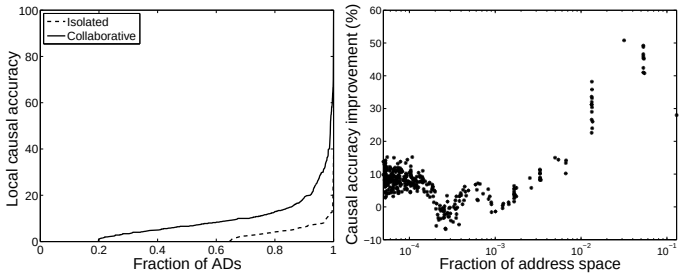


(a) Flow records received (b) Number of ADs contacted

Figure 8: Distribution of the communication overhead in terms of the fraction of total exchanged flow records received and the number of ADs contacted.

investigation scenario, where each AD runs a centralized random moonwalk algorithm on locally available traffic data, with optimally chosen parameters.

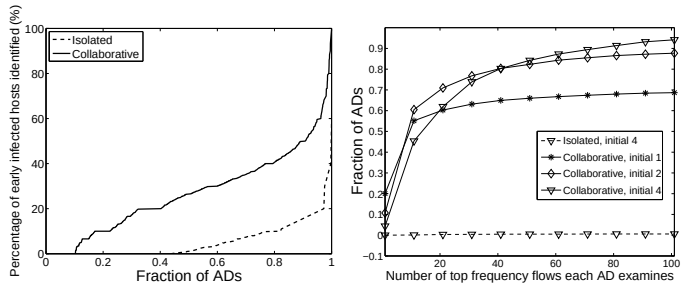
We consider three measures to quantify the perceived local benefit. The first metric is the (local) causal detection accuracy. Figure 9 (a) compares the CDFs of the causal flow detection accuracy across all the ADs with and without collaboration, where each AD returns the top $Z = 100$ after the protocol. With isolated forensics, only 40% of the ADs are able to identify causal flows, while through collaboration more than 80% of ASes can successfully identify causal flows. Overall, we find that there is a substantial improvement in the local detection accuracy through participation over an isolated execution. Figure 9(b) correlates the improvement in the local causal flow detection accuracy (defined as $Accuracy_{LocalView} - Accuracy_{Isolated}$) with the size of the



(a) CDFs of causal detection accuracy

(b) Accuracy improvement vs. size of address space

Figure 9: Causal flow detection accuracy with and without collaboration for each AS ($Z = 100$).



(a) Fraction of initial infected hosts identified

(b) ADs that identified initial causal fbws

Figure 10: The ability to identify the initial infected hosts and the attack entry point with and without collaboration for each AD.

address-space owned by each AD. We observe the larger ADs gain a lot more by collaboration than the smaller ADs.

The other two metrics quantify the ability of each AD to identify the initial infected hosts and the initial attack entry points within its network. Figure 10 (a) depicts the distribution of the accuracy in identifying initial infected hosts. Each AD selects the first 50 internal hosts from among the set of top-frequency flows it has available to it after the distributed protocol [6]. The accuracy of initial host identification is then defined as the fraction of these returned hosts that are among the 50 hosts that are actually infected earliest in time within the AD. We observe that detection accuracy improves across ADs when they participate the NFA.

ADs may also be interested in identifying the first few causal flows where the infection first entered its network. Figure 10 (b) plots the fraction of ADs that successfully identified at least one of the first several entry points, by examining various number of top frequency flows returned with *Local View*. The vast majority of ADs (more than 60%) need to examine only a small number of flows (fewer than 20) before they can detect the first one or two causal flows. These results further corroborate the observations in Figure 9 (a)—collaboration significantly boosts detection performance, and this serves as a strong participation incentive for ADs.

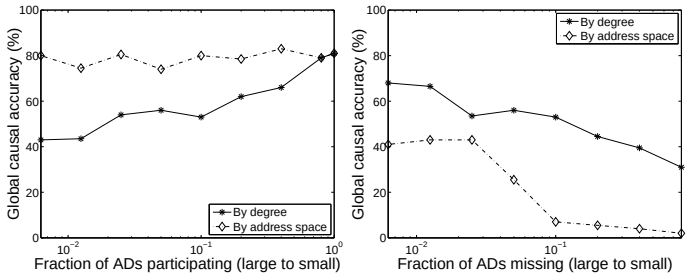
7.1.3 Partial Deployment

There are two natural concerns with incremental deployability: global accuracy and participation incentives. We would like the causal detection accuracy under partial deployment to be comparable to that under complete deployment. Similarly, ADs should be able to observe participation benefits (as defined by the metrics discussed above) even with partial deployment. We consider two partial deployment scenarios, by varying the set of ADs

[6]For many of the smaller ADs, there may not be 50 hosts available to return. In such cases, these ADs simply return all internal hosts that appear along moonwalks

that do not participate based on AD degree and address space size. In each scenario, we consider the case where the k largest domains collaborate and the case where these k largest domains are missing from the NFA, and vary the parameter k.

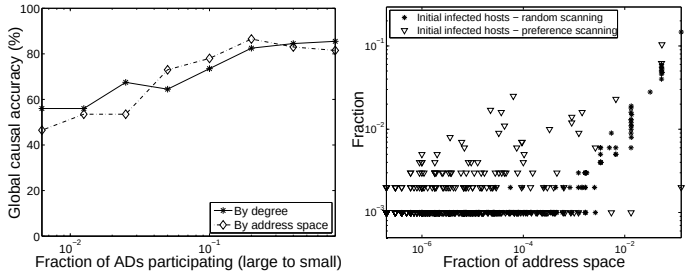
Global Accuracy: Figure 11 plots the global causal flow detection accuracy by varying the number of large



(a) Large ADs participate (b) Large ADs missing

Figure 11: Global detection accuracy under different missing scenarios, with the random scanning worm. We use the best recovery strategy, i.e., *routing horizon*-based recovery.

ADs participating/missing, with the random scanning worm model. We find that when large ADs collaborate, the global causal accuracy is high. In particular, when ADs with largest address space participate in the NFA, the performance is comparable to full deployment. However, when these large ADs are missing from the NFA, the performance degrades significantly, with the global accuracy going down from 80% to 30-40%, even when the vast majority of small ADs collaborate.



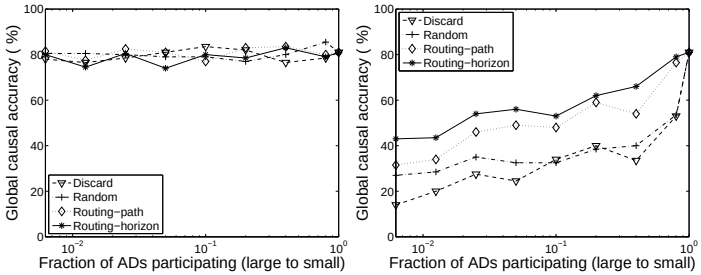
(a) Causal flow accuracy (b) Address space vs. early infected hosts

Figure 12: Impact of worm scanning model on partial deployment performance. We use the best recovery strategy, i.e., *routing horizon*-based recovery.

To investigate this further, we consider the local preferential scanning attack, and show how the global accuracy increases with the number of large ADs collaborating (Figure 12 (a)). Under this worm model, we observe performance degradation when only a small fraction of ADs with largest address space participate. The moonwalk algorithm suggests that the global accuracy is tightly coupled with the initial causal flows in the attack. Hence, we examine the correlation between the AD address-space size and the fraction of initial infected hosts (defined as the 200 hosts that are infected earliest in time in the network) that are actually present within each AD (Figure 12 (b)). For the random scanning worm, ADs with larger address space also observe larger fractions of early infected hosts within their domains, and thus a larger fraction of initial causal flows for the random moonwalk to return successfully. With a local-preferential attack, larger ADs do not necessarily observe many initial infected hosts, hence the corresponding decrease in the number of initial causal flows available to them.

Figure 13 studies the impact of different recovery strategies (discussed in Section 6) on the global detection accuracy, using the random scanning worm. Intuitively, we expect the *Discard* strategy to have the worst performance, and the *Routing-horizon* strategy to have the best performance. However, we find in Figure 13 (a) that when the ADs with large address-space ownership collaborate, the recovery strategies have negligible impact on the overall performance. This is because these ADs already observe the most number of initial infected hosts and a large amount of attack traffic with the random scanning worm model. With only a small number of large degree ADs participating, the recovery strategies do have significant impact on performance (Figure 13 (b)) [7].

The above results suggest that both attack models and recovery strategies may have impact on performance with partial deployment. Further understanding their implications under more diverse partial deployment scenarios is a topic of ongoing work.



(a) By address-space size (b) By topological degree

Figure 13: Impact of recovery strategies on performance, with large ADs participating.

Participation Benefits: Next, we investigate the participation benefits of individual ADs under partial deployment, once again using the random-scanning worm. Figures 14 (a) and (b) show how different partial deployment scenarios affect the participation benefits. We sort ADs based on address-space ownership and degree (from large to small), respectively, and categorize ADs based on their ranks in the sorted order. We then compute the average local causal flow detection accuracy ($Z = 100$) of ADs within each category, and compare the performance achieved by varying the number of large ADs participating. Overall, participation yields benefit for all categories of ADs. Specifically, we find that the large ADs perceive similar benefits to the full collaboration case, even when the rest of the 90% of the ADs do not participate in the NFA. We also observed earlier (Figure 9 (b)) that the perceived performance benefit is greatest for the large ADs in the NFA. These observations bode well for the deployability of the NFA, since it suggests that, for the vast majority of attacks known today which use random scanning for destination selection, domains that have the greatest impact on global performance also have very strong incentives to participate in the NFA.

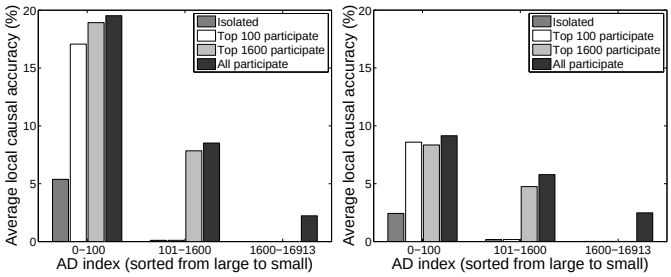
8 Evaluation: Internet2 Study

We use trace-driven simulations to evaluate our architecture, with datasets collected from the Internet2 backbone.

Obtaining traffic data to evaluate federated systems is inherently challenging, due to the lack of access to data from multiple administrative domains. Our strategy is to use data derived from the *Internet2* educational backbone, and map each backbone router into an independent administrative domain to simulate a federated network. The network consists of 11 backbone routers with the topology shown in Figure 15. We refer to each AD in our federation using the city name associated with the backbone router.

We take a one-hour trace from each of the 11 routers, collected at the same time on July 1, 2005. The data

[7] Our analysis using the Routeviews dataset suggests that the correlation between the address-space ownership and AS-degree is not very pronounced



(a) By address-space size (b) By topological degree

Figure 14: Benefi ts of participation under partial deployment, assuming the large ADs participate.

Figure 15: The Internet2 topology, Source: www.internet2.edu

AD (Router)	Hosts (Number of /21)	Traffic volume (Normal/Attack) in million flows			
		Internal	Outgoing	Incoming	Transit
Atlanta (AT)	3295	0.014 / 0.003	0.90 / 0.14	0.53 / 0.13	4.09 / 0.67
Chicago (CH)	25333	0.72 / 0.222	3.55 / 0.82	2.21 / 0.82	1.59 / 0.68
Denver (DN)	17183	0.014 / 0.098	0.78 / 0.67	0.63 / 0.69	4.86 / 0.33
Houston (HS)	1878	0.01 / 0.0011	0.73 / 0.076	0.42 / 0.076	6.04 / 1.01
Indianapolis (IP)	12683	0.06 / 0.0052	1.37 / 0.50	1.09 / 0.51	4.75 / 1.24
Kansas City (KS)	1695	0.004 / 0.0009	0.60 / 0.067	0.43 / 0.069	7.54 / 1.67
Los Angeles (LO)	13848	0.004 / 0.068	2.41 / 0.53	1.90 / 0.51	0.25 / 0.051
New York City (NY)	27189	0.13 / 0.25	2.24 / 0.85	2.34 / 0.87	0.981 / 0.16
Sunnyvale (SN)	2613	0.003 / 0.002	0.34 / 0.11	0.33 / 0.11	0.36 / 0.21
Seattle (ST)	4462	0.02 / 0.006	1.34 / 0.16	3.73 / 0.18	0 / 0
Washington D.C (WA)	9101	0.06 / 0.002	1.75 / 0.394	2.40 / 0.371	1.74 / 0.381

Table 2: Traffi c statistics depicting observed flow counts after our data transformation. In our experiment setup, each router is synonymous with an administrative domain, and each observed unique /21 address maps into a unique host in the federated network.

set consists of sampled unidirectional flow records (with 1 in 100 sampling). IP addresses in the dataset were anonymized by zeroing out the last 11 bits of the source and destination IP address fi elds. We note that the specifi c characteristics of the dataset such as sampling rate and anonymization strategies are dictated by current router technologies and the data disclosure policy of Internet2. Given that our goal is not to perform attack analysis over Internet2, but rather to evaluate a federated architecture using as realistic a workload as possible, we believe this dataset can serve the purpose. Still, there are a number of challenges we have to address to make the traces amenable for our evaluation:

1. Address space ownership: In the federated setting, each domain owns a portion of the address space which it assigns to internal hosts or customers. Since the traces contain only unique /21 addresses, we assume that each observed /21 address maps onto a unique "host" in our federated network consisting of 11 independent domains, and assign each such "host" (i.e. /21 address) to the ADs at which the traffic involving that host was most numerous.
2. Routing: In a federation of ADs, each traffic flow would be routed from the source AD through intermediate ADs to the destination AD, and thus each flow will be logged at multiple domains. To emulate such a federated logging infrastructure, we need to define the routing paths between the 11 ADs in our federation. We use shortest path routing based on hop counts, breaking ties arbitrarily. Once the routes are obtained, for each flow record, we insert the flow record into traces of all the domains on the path between the corresponding source and destination ADs (obtained from the address space ownership mapping).
3. Flow directionality: Our algorithm for attacker identification requires that flows (communication events) have directionality semantics consistent with the directionality of attacks. Rather than artificially impose directionality semantics on the unidirectional flow records, we use protocol and application port specific heuristics: (1) for conservativeness, we use only TCP flows to simulate all the normal traffic generated in the federated network, (2) we look at TCP flags such as the SYN flag seen in the flow record, and (3) we use the application port numbers to infer client-server connections.

Our transformations enable us to emulate a federated network on the Internet2 dataset, so that we can have a first order evaluation to shed light on the effectiveness of the federation. We believe that our algorithm and overall performance characteristics are not biased by these transformations.

We manually inject attack traffic records into the trace. This attack simulates a random-scanning worm, with each infected hosts performs 0.5 scans per second to a randomly selected host in the network. We assume there is 0.1 fraction of hosts that are vulnerable. The attack lasts for 1000 seconds, compromising all vulnerable hosts in 350 seconds. The total unique attack traffic is roughly 22% of the overall unique traffic observed in the network.

Table 2 presents the statistics of the one-hour traces for the 11 ADs after the preprocessing. For each domain, we classify both normal and attack traffic into four categories: "internal" traffic with both the source and destination within the same domain, "outgoing" traffic which originates from source hosts belonging to the domain's address space and intended for destinations outside the domain, "incoming" traffic intended for destination addresses owned by the domain, and "transit" traffic passing through the domain.

For each of the following experiments, we perform 5 independent runs with traffic normalization factor $\eta = 10^{-3}$, the sampling window size $\Delta t = 100$ seconds, and the maximum path length $d(i) = 20$ for each local moonwalk inside a domain. We report the mean observed over the 5 runs. The standard deviation observed across the different runs was negligible, and we do not present these results for brevity.

8.1 Performance Comparison with the Centralized Algorithm

We first compare the performance of the distributed algorithm with the centralized algorithm. We use a slightly modified version of the algorithm described in [5], with two enhancement heuristics: (1) loop-avoidance (walks do not visit the same host twice) and (2) host-dispersal (the selection of initial flows to start moonwalks covers a larger host population instead of being uniformly distributed over the set of flows). The centralized implementation has access to global traffic records, obtained by merging the 11 independent traces and removing duplicate flow records introduced through the routing transformation.

We use two performance metrics introduced in [5]: the causal flow detection accuracy and the attack flow detection accuracy. The detection accuracy is defined as the fraction of the top Z returned flows that are actually causal (or attack) flows. With the distributed algorithm, these Z flows are identified using the "Global Merge" procedure presented in Section 5.2. In Figure 16, we observe that with all 11 domains participating, the detection accuracies achieved by the distributed algorithm after 15 iterations approximates the accuracies of the centralized algorithm. We also show how the performance approaches that achieved by the centralized

algorithm as the algorithm proceeds over multiple iterations in Figure 17. Initially, the algorithm is more successful at picking attack flows. The reason is that the first step of the moonwalk algorithm involves starting walks at random flows, and the attack flows are inherently more numerous than causal flows. As the algorithm iteration increases, the majority of the returned flows are causal flows, and the detection accuracies for both causal and attack flows converge after 10-12 iterations.

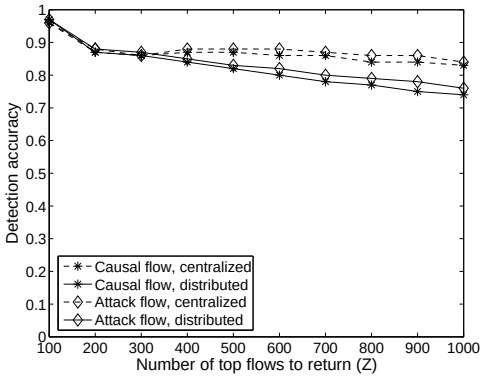


Figure 16: The accuracy achieved by the distributed algorithm compared with the accuracy achieved using the centralized implementation by varying the global top frequency flows (Z) to return

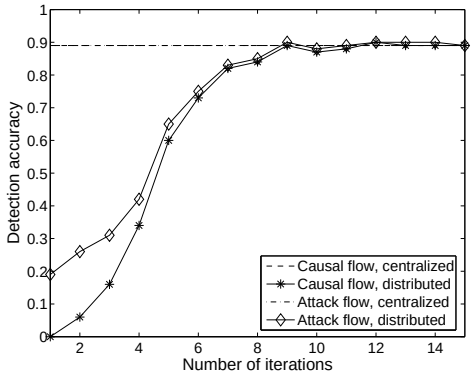
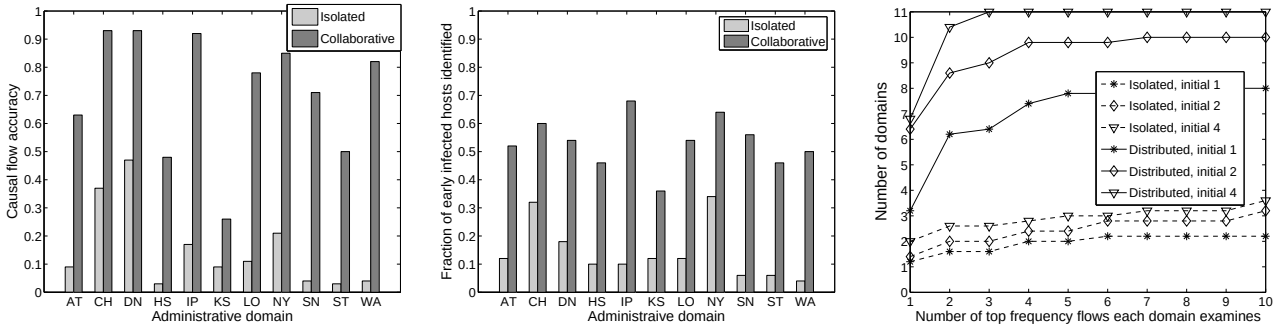


Figure 17: The detection accuracy achieved by picking the top $Z = 200$ flows after each cross-domain message exchange compared with centralized detection accuracy. For the centralized algorithm, the detection accuracy of the causal flows and the accuracy of attack flows achieved are the same.

8.2 Benefits of Participation

For a federated architecture to be viable, each participating domain must receive benefit for participation. We first consider the case where each participating domain jointly performs distributed random moonwalks, but does not participate in the final *Global Merge* operation, to avoid sharing traffic flow information. Each of them just relies on the *Local View* after distributed moonwalks. Thus, throughout the algorithm and the information sharing steps, each domain can only access the traffic records it has observed locally. We examine the three measures proposed in Section 5.1 for quantifying the perceived benefit of participation, and compare the performance achieved with the *Local View* against the best performance that can be achieved in isolation (i.e., centralized random moonwalks on local traffic data in isolation, with optimally chosen algorithm parameters).

We first examine the causal detection accuracy, shown in Figure 18 (a). For all the domains, we find



(a) (b) (c)

Figure 18: Performance with and without participation for each participating domain: (a) Causal flow detection accuracy (b) Fraction of initial infected hosts identified (c) Number of domains that successfully identified the actual initial causal flows

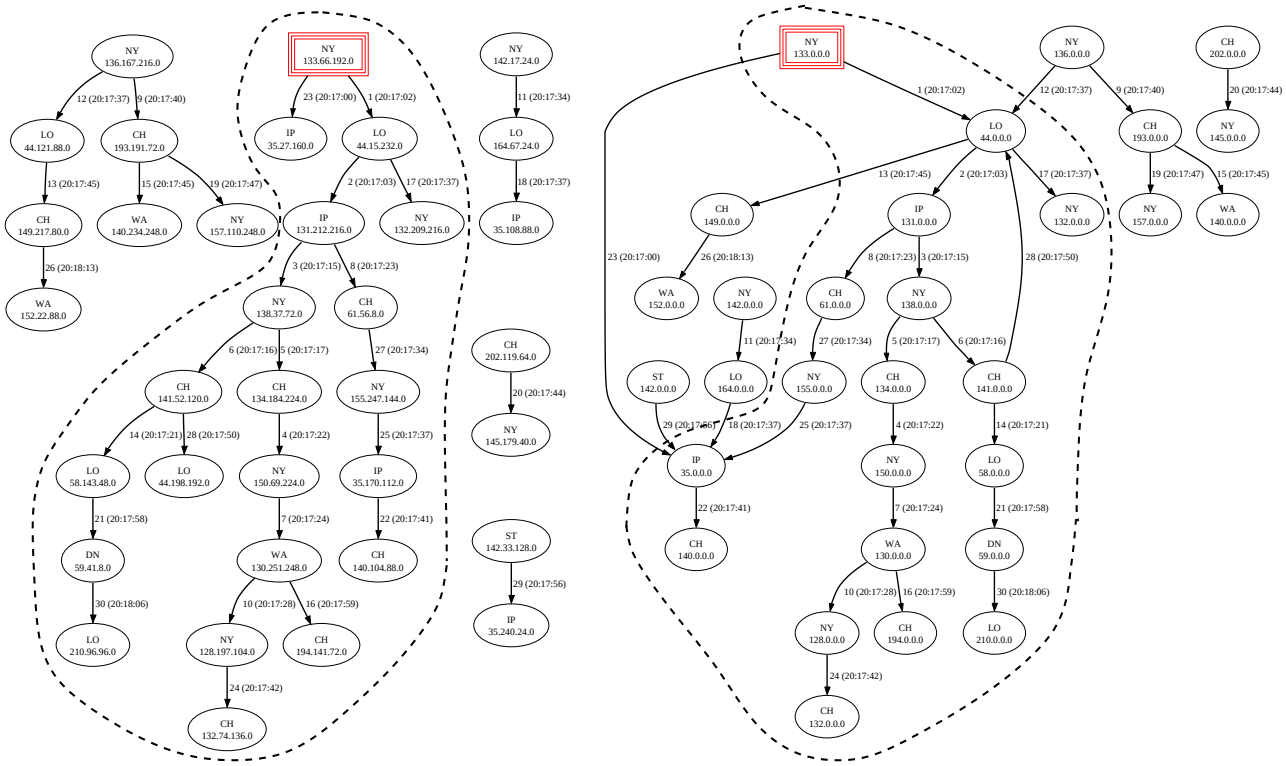
that there is a substantial improvement in the local detection accuracy through participation over an isolated execution. For example, the performance of domain *IP* improves from around to 20% to 80%. Even though the absolute accuracy is relatively low for domain *KS* compared with other domains after collaboration, the relative performance increase is at least 80%.

The second measure we have discussed is the ability to identify the set of hosts that are infected earliest in time. Each domain identifies the first 50 internal hosts by examining the top frequency flows in order. Figure 18 (b) shows the fraction of these 50 hosts which are among the 50 internal hosts that are actually infected earliest in time for each AD. We observe that compared with isolated execution of random moonwalks, the ability to find the initial infected hosts increases significantly with collaboration.

We then evaluate the third local benefit measure in terms of the ability to find the entry point(s) of the global attack within each AD's perimeter. In Figure 18 (c), we examine the number of the 11 participating domains that successfully identified one of the first several entry points, by looking at a small set of top frequency flows returned at each local domain. The X-axis is the number of top frequency flows that each domain will investigate. The Y-axis is the number of domains that identified at least one of its initial causal flows. We find that by looking at 5 top frequency flows, a majority of all the domains will be able to identify the first causal flow, at which the attack first entered that domain. By looking at the 3 top frequency flows, all participating domains will successfully identify at least one out of the first 4 initial causal flows. In comparison, we observe that only a very small set of domains will be able to identify the attack entry points correctly in isolation.

We investigated why certain ADs failed to identify the exact initial causal flow. In particular domain *AT* is an interesting instance, as our algorithm consistently failed to discover the initial causal flow within the domain. We found that the destination host of the first causal flow has a much higher *fan-in*, in terms of the number of incoming flows, than the destination hosts of the top frequency flows returned by the algorithm. Hence the random moonwalks did not assign a high rank to the causal flow that infected this host, at which the attack first entered domain *AT*.

We proceed to study the benefit of each AD in terms of the knowledge of a global initial attack propagation graph. Figures 19 and 20 show the structures of the global initial host contact graph that will be reconstructed at each participating domain, by sharing the results using the "Global Merge" protocol with different levels of host anonymization. In this case, each AD contributes its top $Z = 30$ local flows that originated from the domain to identify the global top 30 flows. Each node in the figure represents an IP prefix from a domain, and edges represent communication flows between hosts. Even with /8 address anonymization, we observe that the tree-structured nature of the attack is imminent in the reconstructed graph. The hosts sitting at the top levels of the trees provide a small candidate set for identifying the actual worm origin. The true source of the attack (marked as the square node), which we artificially injected into the traffic trace, is located inside the *NY* domain



(a) With /21 host IP addresses (b) With /8 host IP addresses

Figure 19: The structure of the flow graph constructed with different degrees of host anonymization using the 30 top frequency flows on a global scale ($Z = 30$).

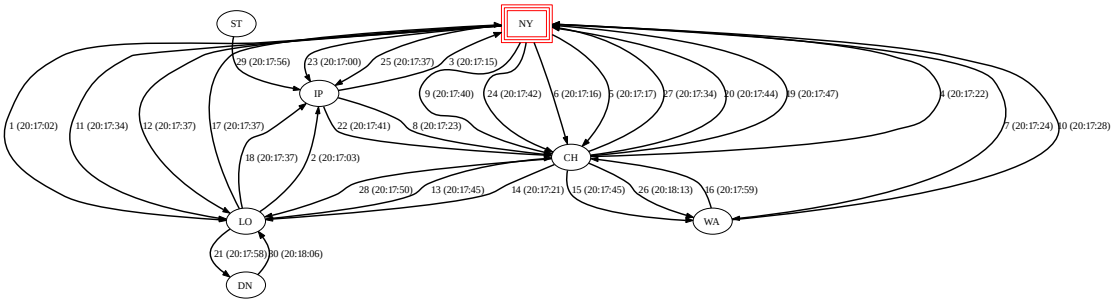


Figure 20: The structure of the flow graph constructed without host IP addresses (i.e., replace the host identity with domain names) ($Z = 30$).

with the largest tree-branching structure below it. We circle this largest tree structured flow subgraph using dashed lines on both Figure 19 (a) and Figure 19 (b), to show the correspondence.

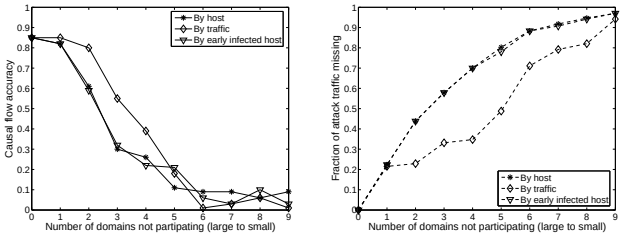
8.3 Partial Deployment

In practice, such a federated architecture will be incrementally deployed, with some portions of data initially missing, and some ADs initially unwilling to participate. We construct three potential partial deployment scenarios, by varying the set of ADs that do not participate:

1. Traffi c – We assume that the ADs with the most (least) observed traffi c participate.
2. Host – ADs with the most (least) address space ownership participate.

3. Early infection – ADs with the most (least) number of early (considering the first 200 infected hosts) infected hosts participate.

We first study the causal flow accuracy degradation (with $Z = 200$) in a worst case partial deployment scenario, where large domains do not participate by removing ADs in decreasing order of traffic (host/early infection) from the federation. For example, for traffic-based removal, we assume that the ADs with the highest total traffic volume do not participate. We use the *routing-based* recovery strategy for handling partial deployment as presented in Section 6.2. We observe in Figure 21(a) that with a small number of (1-2) large domains not participating, we can still achieve a reasonably high detection accuracy, but the performance degrades significantly (to less than 20% accuracy) with 5 or more large ADs not participating. Removing ADs by early infected hosts has similar impact on performance to host-based scenario, both of which give consistently worse performance degradation than traffic based participation. To investigate the reason, we show in Figure 21 (b) the fraction of attack traffic that is available to the distributed algorithm for the different scenarios. We see that with both the host-based and early infection based case, the fraction of total distinct attack traffic missed is consistently larger than the traffic-based scenario. With 5 or more large domains not participating, more than half the total attack traffic is not available to the federated monitoring infrastructure. Since our attack is a random scanning worm, and the distribution of vulnerable hosts is uniform across the host-space, ADs with larger number of hosts are more likely to have more early infected hosts as well. These ADs will thus observe a larger amount distinct attack traffic compared to ADs that may route high volume of traffic but have smaller host populations.

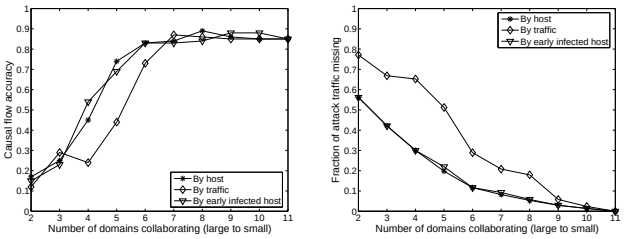


(a) (b)

Figure 21: (a) Causal flow detection accuracy with number of domains not participating (b) The fraction of attack traffic missing vs. the number of domains not participating

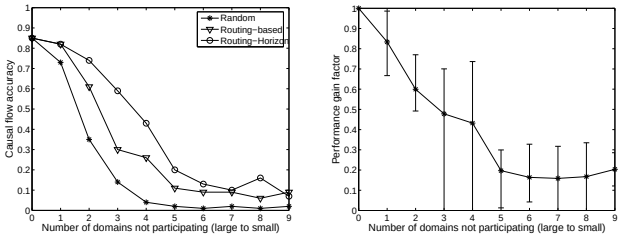
The next question we look at is how much performance the federation can achieve with several large domains participating. We show in Figure 22 (a) the causal flow accuracy by varying the number of largest ADs participating. We find that even 50% of the federation with 5 or 6 small ADs not participating, the performance is close to that achieved in a complete deployment scenario. Again, we also plot the fraction of distinct attack traffic missing in Figure 22 (b) for such a scenario. For reasons we explained previously, the participation of ADs with larger number of hosts and early infected hosts tend to achieve better performance than the participation of ADs that observe more traffic.

We proceed to study the effectiveness of different recovery strategies that we proposed in Section 6.2 to handle partial deployment. Figure 23 (a) shows the causal flow accuracy, assuming that the domains with the largest number of hosts are not participating. With only one domain missing, all strategies have similar performance. As the number of non-participating domains increases, the broadcast-based strategy achieves consistently better performance, since it has a better partial view for continuing the distributed moonwalk. The routing-based strategy outperforms random selection. We also observe that on our 11-node federation, the difference in the performance between the relatively light-weight routing-based recovery strategy, and the heavy-weight broadcast strategy is not substantial. Evaluating the effectiveness of these recovery strategies on larger federated scenarios is a topic of ongoing work.



(a) (b)

Figure 22: (a) Causal flow detection accuracy vs. number of large domains participating (b) The fraction of attack traffic missing vs. the number of large domains participating



(a) (b)

Figure 23: (a) Causal flow accuracy with different recovery strategies to handle partial deployment (b) Local benefit with large domains not participating

Finally, we investigate the question of how much benefit each domain can receive with partial deployment, compared to an ideal deployment scenario. We define a *performance gain factor* to quantify the relative performance increase as:

$$\frac{\text{Accuracy}_{Partial\ deployment} - \text{Accuracy}_{Isolated}}{\text{Accuracy}_{Global\ deployment} - \text{Accuracy}_{Isolated}}$$

Figure 23 (b) shows the maximum, mean, and minimum performance gain factor for the participating domains with various number of large domains (host-based) not participating. On average, the relative performance gain decreases linearly when the first several large domain do not participate. When 4 large domains do not participate, the average performance gain for the participants is close to 50% of what could be achieved with complete deployment, suggesting that most ADs gain benefit and have incentives to participate even in partial deployment scenarios.

Our evaluations with different partial deployment scenarios suggest that the global performance degrades gracefully with missing data, and that participants receive benefits from collaboration even with partial deployment. The non-participation of a large number of smaller (stub) domains that do not have large traffic or address-space has negligible impact on the global performance. Further, the federation can achieve reasonable good causal flow detection accuracy even if one or two largest ADs do not participate, but the accuracy degrades significantly if a majority of the large ADs do not participate.

9 Limitations

We discuss limitations and vulnerabilities in our design, and outline potential remedies. Formulating realistic adversarial models and strategies, and devising more comprehensive solutions to such attacks, are topics of ongoing work.

Slow-rate attacks: Attackers may wish to evade detection by slowing down the attack propagation rate to blend their propagation into the normal background traffic. Our initial results in [5] suggest that even slow propagating attacks exhibit "tree" like structured traffic pattern. While the detection accuracy of our approach decreases with a lower attack scanning rate, the effectiveness of these slow-rate attacks in infecting all vulnerable hosts is reduced as well. Future work includes further investigating the detection accuracy of the NFA

across a wider spectrum of attack propagation rates.

Server-targeted attacks: Attackers may also target servers as initial victims for spreading epidemic attacks. Since servers usually have a large number of incoming flows, their presence in the attack-tree could help conceal the attack tree structure, especially during the early spreading stage. In particular, once random moonwalks reach outgoing flows from a server, the probability of the walks converging to the malicious incoming causal flow becomes smaller. However, servers are generally better engineered and protected, and often operate in configurations that are not likely to be susceptible to client software vulnerabilities. Hence, we expect such attacks to be more difficult in practice.

Latent-period varying attacks: In our protocol, participating ADs use a bootstrap mechanism to agree on a sampling window Δt as the maximum look-back window for selecting flows in the distributed moonwalks. Attackers may vary the latent period between the time a host gets infected and the time the host starts scanning to defeat the system. To detect such attacks in forensic analysis, ADs may choose a slightly larger Δt, to identify attacks with varying latent periods. ADs may also adaptively select Δt at various steps along the moonwalks, for example, choosing an Δt that yields at least one incoming flow for continuing the moonwalk.

Topology-aware attacks: Our experiment results of different worm models with partial deployment suggest that the attackers may use the knowledge of network topology and the NFA's deployment status to evade detection. With only a subset of ADs participating, an attacker can restrict the spread of the attack during the initial stage within the network boundary of non-participating ADs. Only after a sufficient number of hosts have been infected does the attacker launch a full-fledged attack starting from these "seed" hosts that have been infected initially. Since random moonwalks are most effective in identifying initial causal flows, the accuracy with which causal flows are detected might degrade in such scenarios. However, even in such cases, we expect that the suspicious flows identified by our system will point toward the (non-cooperating) ADs from which the attack has originated, for further out-of-band investigation.

Information leakage attacks: Due to economic and privacy considerations, ADs may not wish to reveal their internal topology, traffic engineering policies, and internal traffic patterns, especially to competitors. A federated architecture raises the possibility of traffic inference attacks, either launched by competitors with vested interests or via accidental information leakage. For example, a strategic adversary may be able to craft traffic patterns or flow traversal counts that maximize the information leakage from potential victims during the distributed protocol. However, due to the inherent randomization in the moonwalk algorithm, we believe such attacks to be difficult, if not effectively impossible.

Mis-information and information-hiding attacks: A malicious participant may inject false information, or refuse to divulge information relevant to the investigative effort.

In a *false implication* attack, a malicious participant attempts to forge results that implicates an innocent AD (or host) as being the source of an attack. We are currently investigating methods to detect and mitigate such attacks, assuming that a sufficient number of non-malicious ADs participate. The intuition is that with the majority of "honest" participants, communication flows between infected hosts and their eventual counts after the distributed protocol will be exposed even if certain domains attempt to hide the information.

One possibility, is that a participant may choose to delay revealing its counts to others until it receives their counts in the *Global Merge* procedure. Based on the information received from others, it can potentially modify its own flow reports to generate false counts. To prevent generation of false counts using other domains' input, we can use secure commitment protocols (e.g. using a cryptographic-hash of its contents) to bind each participant to the result they are required to report.

10 Conclusions

We proposed a protocol for performing distributed random moonwalks, which can enable participating ADs to identify local attack entry points, and can provide additional functionality to pinpoint global attack origins. Our design is suitable for deployment in an Internet-like federation for three reasons. First, the protocol operates with limited information disclosure, without requiring participants to reveal traffic records to other participants that would otherwise not be available. Second, participating domains realize enhanced local attack investiga-

tion, and thus receive substantial incentives for cooperation. Third, the framework is incrementally deployable and can handle non-participation and missing data gracefully. We believe that our design and results provide a technical basis of a Network Forensic Alliance (NFA), a collaborative effort involving multiple ADs to provide network-wide and localized forensic capabilities.

11 Acknowledgments

We gratefully acknowledge the assistance of Internet2 for providing us with access to their traces. We also thank Sachin Katti and Andy Myers for their comments and suggestions for improving the paper. This work is sponsored in part by the NSF SAFE project under Grant CNS-0433540 and NSF 100x100 project under Grant ANI-0331653, and in part by the US ARO Dragnet project under Grant DAAD19-02-1-0389.

References

[1] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast Portscan Detection Using Sequential Hypothesis Testing," in *Proc. of IEEE Symposium on Security and Privacy*, 2004.

[2] D. Moore, C. Shannon, G. Voelker, and S. Savage, "Internet Quarantine: Requirements for Containing Self-Propagating Code," in *Proc. of IEEE INFOCOM*, 2003.

[3] H. A. Kim and B. Karp, "Autograph: Toward Automated, Distributed Worm Signature Detection," in *Proc. of 12th USENIX Security Symposium*, 2004.

[4] "Fingerprint Sharing Alliance," http://www.arbor.net/fingerprint-sharing-alliance.php.

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

[6] "The Internet2 Ablilene Network," http://abilene.internet2.edu.

[7] Y. Zhang and V. Paxson, "Detecting Stepping Stones," in *Proc. of the 9th USENIX Security Symposium*, 2001.

[8] D. L. Donoho, A. G. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford-Chen, "Multiscale Stepping-Stone Detection: Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay," in *Proc. of The 5th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2002.

[9] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," in *Proc. of ACM SIGCOMM*, 2000.

[10] H. Burch and B. Cheswick, "Tracing Anonymous Packets to Their Approximate Source," in *Proc. of USENIX LISA Systems Administration Conference*, 2000.

[11] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer, "Hash-Based IP Traceback," in *Proc. of ACM SIGCOMM*, 2001.

[12] A. Kumar, V. Paxson, and N. Weaver, "Exploiting Underlying Structure for Detailed Reconstruction of an Internet-scale Event," Proc. of Internet Measurement Conference, 2005.

[13] M. A. Rajab, F. Monrose, and A. Terzis, "Worm Evolution Tracking via Timing Analysis," in *Proc. of Workshop on Rapid Malcode (WORM)*, 2005.

[14] J. Bethencourt, J. Franklin, and M. Vernon, "Mapping Internet Sensors with Probe Response Attacks," in *Proc. of the 14th USENIX Security Symposium*, 2005.

[15] Y. Azar, A. Fiat, A. Karlin, F. McSherry, and J. Saia, "Spectral Analysis of Data," in *ACM Symposium on Theory of Computing*, 2001.

[16] S. Acharyya and J. Ghosh, "Outlink Estimation for Pagerank Computation under Missing Data," in *Proc. of the 13th International World Wide Web Conference*, 2004.

[17] Z. Mao, J. Rexford, J. Wang, and R. Katz, "Towards an Accurate AS-Level Traceroute Tool," in *Proc. of ACM SIGCOMM*, 2003.

[18] "The Route Views Project," http://www.routeviews.org.