A Functional Reference Model of Passive Network Origin Identification

Thomas E. Daniels
Iowa State University Information Assurance Center and
Department of Electrical and Computer Engineering*
3222 Coover Hall
Ames, IA 50011
daniels@iastate.edu

July 22, 2003

Abstract

Determining the originating node of network traffic is a key problem in network forensics. As it is unlikely that a network attacker will leave direct evidence of his identity, it is useful to find his point of entry into the network. This, along with further host–based investigation, can tie a given suspect to an attack. Past work at this origin identification problem has assumed cooperative users (authentication), simple mechanisms of origin concealment (i.e. correlating spoofing or island hopping alone), modifying network protocols (traffic marking), or host-embedded protocols (i.e. Carrier's STOP protocol). As this work is usually specific to a single type of origin concealment, we know little in general about the origin identification problem. In this paper, we discuss passive approaches that do not modify traffic, but rather, they store observations for later analysis.

We present a general reference model of passive origin identification. The reference model defines the features of all passive origin identification systems known. It is a functional model as it defines the components in terms of their general behavior and goals. The reference model is useful for reasoning about the behavior and flaws of origin identification systems. The model is also quite useful for discussing and teaching origin identification techniques.

The model leads to several necessary and sufficient conditions for some level of passive origin identification in general. The first is separation of the network by monitors. The second is sufficient storage to permit later analysis. Furthermore, the model leads to several additional mutually sufficient conditions for passive origin identification in general. These are accurate correlation of traffic outputs to corresponding inputs and a trusted communication path between the analysis agent and the network monitors. By examining each of these conditions in the context of their applicability to forensic evidence, we suggest that passive origin identification will remain a preliminary investigation tool unless monitors are widely deployed in network hosts.

This work is the first general theoretical framework for network forensics using passive monitors. It facilitates comparison of origin identification techniques and suggests new problems facing the field.

*Portions of this work were supported by National Science Foundation Grant EIA-9903545 and by the sponsors of the Center for Education and Research in Information Assurance and Security and the Iowa State University Information Assurance Center

1 Introduction

1.1 Network Assumptions

We model a network as an *observed network* of the form $G = (V, E, IM, XM)$ where $IM \in V$ and $XM \in E$. V and E are nodes and edges of the network which we assume is undirected. IM are those nodes that are monitored internally and XM are those edges which are monitored by the NOIS.

A node, $v_0 \in G.V$ may generate an NDE, m, which traverses a sequence of nodes, $[v_0, ..., v_k]$ where $(v_i, v_{i+1}) \in G.E$ for $0 \leq i \leq k-1$. An input m traverses a node v_i if it causes v_i to output an m'. This is modeled by a relation $v_i.T(m, m')$.

The overall goal of passive origin identification is given an observation of m_j at v_j, determine the origin set of m_j with the evidence available from the monitor. The goal of superset origin identification is given an observation of m_j at v_j, determine the smallest set of nodes that could contain $origin(v_j)$ given the evidence available. We call it superset origin identification because we are looking for a set of nodes that is a superset of $origin(v_j)$. The goal of approximate origin identification is to find some arbitrary v_i where $0 \leq i < j$ such that an ancestor of m_j traversed.

1.2 Passive Network Origin Identification

Passive NOISs monitor the network in one or more locations but do not modify the NDEs observed. A passive NOIS might monitor the network at several points and gather the results at one place for analysis, or it might monitor at a number of nodes in the network, storing observations for later queries. A number of passive systems (or proposals for them) have been published [Row99, SDS00, CD97, MOT+99].

Passive NOISs may monitor internally or externally. Internal monitoring is done inside a network component that processes an NDE. An example would be a system of routers that can be made to log packets to trace a denial of service attack such as DoSTracker [CD97]. External monitoring is done using access to network media by a device that does not forward traffic. A NOIS that relies on external monitoring is described by Staniford [SC95].

In Figure 1, we have seven network components with four monitors in place. M1, M3, and M4 are external monitors, and M2 is an internal monitor in component N2. The figure depicts the actual network topology and layout of monitors in the network. A monitor is an actual device that records or makes observations of network traffic available for querying. It is important that we define the monitor here as we will later use an abstraction called "the observer" to reason about and aggregate observations of traffic.

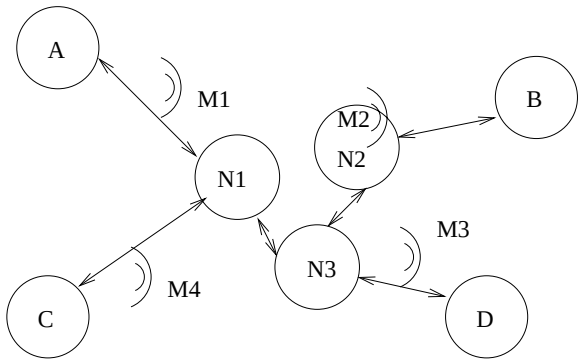

Figure 1: A network with three monitors installed. M1, M3, and M4 are external monitors. M2 is an internal monitor in the network component N2.

An external monitor directly observes the network media. The external observer can observe units of traffic crossing the network media to record their value, location, and time of detection. Depending on the media type, the external monitor may also be able to determine directionality and signal characteristics such as line voltage. For instance, in a broadcast network, it may be difficult for the external observer to determine the component that put the NDE onto the media and likewise which component(s) will process the NDE. Alternatively, if the media is strictly point–to–point, this information is available assuming the directionality of the NDE can be de-

tected. The ease with which media may be externally monitored depends on the type of media. For instance, wireless networks such as 802.11b can be monitored with a receiver that is in range of transmitters, whereas monitoring a fiber optic cable will require physical access [Tan88].

The features of an NDE that an external monitor can observe are as follows: contents, headers, times of observation, and signal qualities. We can represent an observation of an NDE, m, as $(m.h, m.c, m.t, m.s)$. We typically neglect the $m.s$ as signal qualities are unlikely to be available or useful.

An internal monitor can record observations inside a network component transferring NDEs. An internal observer can observe the NDEs on all edges adjacent to it. Furthermore, an internal monitor may also detect NDEs created, received, or dropped by the component. An internal monitor is potentially more powerful than an external one because the internal monitor may determine the corresponding output NDEs caused by the inputs. This may require significant modification of the network component, including modifications to the operating system or hardware. If an internal monitor cannot detect transformations, generation, or dropping of NDEs, it is equivalent to external monitors on all edges adjacent to the internally monitored node.

2 Relays: Modeling Nodes and Subgraphs of Networks

To discuss network behavior more easily, we now introduced the notion of the relay as a mechanism for discussing origin concealment systems and their components. As the goal of an origin identification system is to determine the identity of the origin of network traffic despite efforts at origin concealment, we will use the relay abstraction here.

A relay can model a single node from the network or a connected component of the network. The relay, N, has a set of possible inputs, $N.in$, and outputs $N.out$, and some transform relation $N.T()$ that pairs input NDEs with output NDEs. It also can generate, $N.G(t, S)$, or drop traffic.

For origin identification purposes, we usually discuss observed relays. An observed relay is one with all edges connecting it to the rest of the network monitored. Given an observed network, $G = (V, E, IM, XM)$, we can then define an observed relay. N is an observed relay if it is a subgraph of G and $\forall (u, v) \in G.E, (u \in N.V \wedge v \in (G.V - N.V) \rightarrow ((u, v) \in XM \vee u \in IM \vee v \in IM))$. In other words, every edge between the subgraph and the rest of the graph is covered either by an external monitor or an internally monitored node incident with the edge.

We can then define a contraction of the original graph such that every node in the new graph is an observed relay. We call this the *edge observed graph* as the edges corresponding to an edge in the edge observed graph are observed by a monitor. Simple versions of these graphs are when $IM = V$, all nodes internally monitored, and $XM = E$, all edges externally monitored.

An edge observed graph, $G' = (V', E', IM', XM')$, can be computed for a connected network, G, by finding connected components. First, let $V' = IM' = IM$ as all internally monitored nodes are observed relays. Next, form a graph $G_{tmp} = (V, E - XM - \{(u, v) \in E : u \in IM\})$. G_{tmp} has all edges monitored by internal and external monitors removed. Find the connected components in G_{tmp}. For each connected component in G_{tmp}, add a node to V'. If two connected components in G_{tmp} shared an edge in G, add an edge to E' between the corresponding nodes. Finally, because G is connected, the edges between the connected components were all monitored. Hence, the externally monitored edges in the edge observed graph are those in E' not adjacent to an internal monitor so that $XM' = E' - \{(u, v) \in E' : u \in IM'\}$.

In forming an edge observed network, we may logically merge multiple external monitors or have ignored external monitors. The merging of monitors occurs when two connected components from G_{tmp} share monitored edges in the original graph. The result is that the two edges are merged into one in G'. We discuss observers

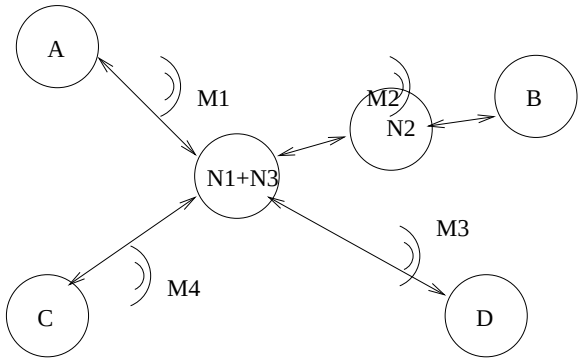

Figure 2: An edge-observed contraction of the network shown in Figure 1. $N1$ and $N3$ were contracted into a single relay. All other nodes in the original network were observed relays already.

below that allow us to model the merger of monitors. An edge observed graph may not incorporate a monitored edge from the original graph if the edge's removal does not take part in partitioning the graph into components. An example of this would be a single monitored edge in a biconnected graph. Any edge's removal does not partition the graph.

3 Observers

An *observer* is a abstraction of one or more monitors combining subsets of the data captured by the monitors. In general, an observer is formed by selecting some subset of observations from one or more monitors while appending each observation with the an identifier of the monitor that made the observation.

Given a relay modeling an edge observed relay, N, we can define an input observer by selecting all inputs observed incoming on edges to N by each monitor on edges incident with N. Similarly, we can form an output observer for N. The result is that we can form a relay with observed inputs, $O.in$, and observed outputs, $O.out$.

An internal observer for a single node is simply the observations from the node's internal monitor with an identifier for the node appended. Figure 4 shows an internal observer and the data it can collect.

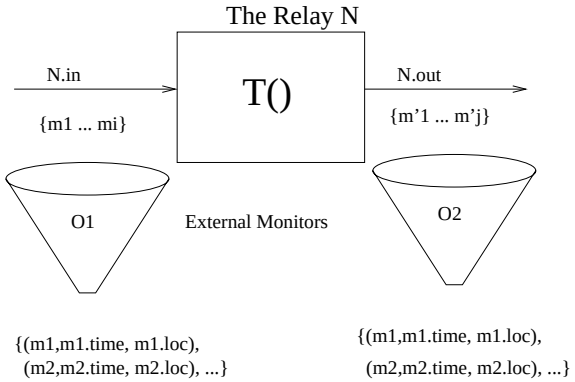

Figure 3: Functional diagram of a network component with external observers at its inputs and outputs.

Figure 3 shows a model of an external origin identification system around a relay, N, with two observers, O_1 and O_2, and an NDE correlation function. O_1 is an input observer representing the inputs of one or more actual monitors. They could be implemented as many distinct monitoring devices on many network media connected to nodes represented by N or only one device monitoring a single full duplex connection to N.

Input/output edge refers to the connection where the NDE was observed. This could be denoted by one of N's interface names or an incoming point to point media name. We also assume that the NDE can be observed as being incoming or outgoing with respect to N. These can be represented as hardware layer identifiers in the NDEs identifier vector.

Figure 4 shows a network relay N with inputs and output NDEs as in Figure 3. In this case, $O1$ monitors N internally and produces data in addition to the external case. The relationship between NDEs input and output may be determined directly. In the figure, the NDEs generated by N are reported as well as those dropped.

The availability and quality of this relationship information depends on a number of factors. If

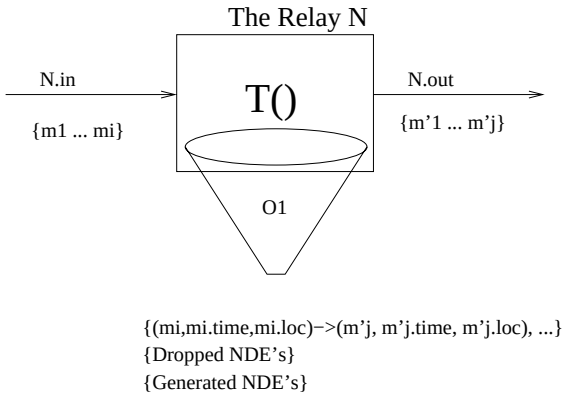

Figure 4: Functional diagram of a network relay with an internal observer. O1 can observe the input NDE, the time of input, and the receiving interface. O1 can also observe output NDE, the output time, and interface, but more importantly, O1 may correlate inputs and output NDEs. The observer can also detect NDEs dropped and generated by N.

the observer can be tightly integrated into N's software or hardware, then the accuracy of the relation could be high. In DoSTracker [CD97], built-in debugging facilities in routers are used to provide accurate relationship information. If the behavior of N can be simulated, an internal observer could also provide information about the state of N and its inputs so that this dependence information could be created by simulation. This might require a less tightly integrated observer than above and would require further processing of another entity to simulate N.

When N is an arbitrary connected component of the network, the concept of external monitoring is clear, but it is not yet defined for internal monitoring. External observers only require access to the media connecting N to the remainder of the network, so it does not matter whether N is a network or a single node for making external observations.

We define internal monitoring of a relay based upon the possible observations for internal monitoring of a single node. First, assume the network is N as described in Figure 3. Our requirements are that we can observe incoming and outgoing NDEs, their dependency relationships, and which of them were dropped or generated by N. We note that this says nothing of the NDEs that never leave the boundary of N. We will justify this in an upcoming section.

To achieve internal monitoring for a relay of more than one component, a necessary condition is that N is an observed relay. A second necessary condition is the ability to determine the dependence between incoming and outgoing NDEs. We must also be able to infer that NDEs are dropped or generated in N. We call the problem of determining the dependencies between NDEs, *NDE correlation*, and we will describe it in greater detail later.

There are situations where a monitor may not be necessary on every edge to internally monitor a relay. Knowledge of a network might give information that certain edges could not carry the type of NDE in which we are interested nor carry their ancestors. In this case, we could simply eliminate the edges from the graph for that type of NDE. An example of the situation might be a well-secured email gateway between a classified and unclassified network. Even though there is a connection, traffic other than email may not pass through the gateway, so if the NDEs of interest to the NOIS are neither email nor related to email then the gateway need not be monitored.

4 Functions of a Network Monitor

A network monitor collects observations and may store them for later analysis or retrieval. The internal processing of a network monitor can be represented by functional components for control, selection, data reduction, storage, and reporting. Figure 5 illustrates these components and how they interact.

The control unit of a monitor allows a remote entity to send it commands. In some cases, such as DoSTracker [CD97], the process of making observations may interfere with the functioning of the device by consuming processing power, I/O

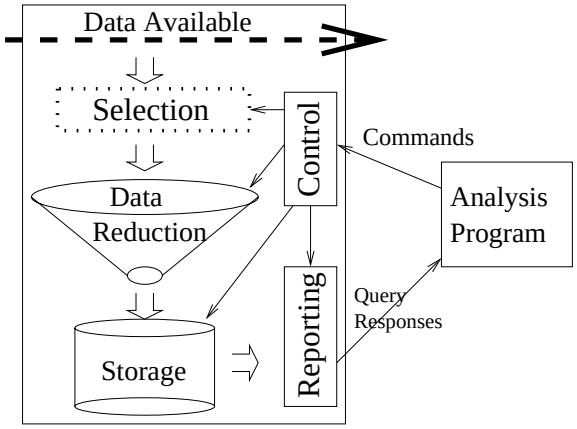

Figure 5: The diagram shows the functional components of a passive network origin identification system. The box to the left contains a network monitor with its functional units.The box to the right is an analysis system that sends commands to the monitor, and receives data from the monitor to trace an NDE.

bandwidth, or delaying NDEs further. This is in addition to possible storage used to store the observations. Because of this, a monitor may have a mechanism for beginning, ending, and configuring the observation process so that resources can be devoted to the normal functioning of the device. The control functions are also responsible for authenticating commands to the monitor to prevent unauthorized access. Other commands for retrieving observations are passed on to the reporting unit.

The selection unit in a monitor takes commands from the control function. The selection functionality extracts NDEs from all the NDEs observable by the monitor. Commands from the control unit can instruct the selection unit to use either event-based or random sampling or both to select NDEs from those observable.

Event-based sampling is the method used in the past work. In event-based sampling, a monitor awaits an event to trigger the taking of an observation. Events include reception of an NDE matching some criteria, a change in state of the monitor, or specific commands from the control unit. Random sampling would be the alternative to event-based sampling. In random sampling, the monitor would select an NDE for observation based on some probability distribution. We know of no instances of this among the past work in passive NOISs, but random sampling for marking has been used in active flow marking schemes[SWKA00a, Bel00, SWKA00b].

DoSTracker's [CD97] selection unit is the router's own filtering mechanism set to raise an alarm when a packet matching a signature is input. SPIE's [SPS+01] selection unit is basically the identity function as it monitors every IP packet input. Some types of monitors in IDIP [Row99] use a network intrusion detection system's pattern matching system as a selection unit.

Data reduction is a function of monitors that has been the focus of past work [SCH95, SC95, SPS+01]. Because the amount of storage available to a monitor is limited, it will eventually fill. Data reduction techniques attempt to store reduced yet still characteristic data about the NDEs selected by the selection unit. SPIE [SPS+01] stores hashes of IP packets using a Bloom filter, which later allows recognition of the hash without actually being able to recreate it. Traffic thumbprinting [SCH95, SC95] reduces time slices of TCP streams to counts of selected characters from the streams. Compression techniques could also be applied in the data reduction unit.

The data reduction unit may not be present in all NOISs, as in some cases all selected NDEs may be stored in total. A case where this could be appropriate is in a low bandwidth network where in-depth manual analysis is to be performed on the traffic. In this case, it could be difficult to choose portions of the NDEs to discard because it may not be apparent which portions of them will be useful for analysis.

Depending on its use, a monitor can have a great deal of storage for observations or little. As indicated before, there is a relation between the history length and the amount of storage available to the monitor. The history length, h, is the amount of time that a monitor can

continuously take and store observations without discarding any observations. We can compute $h \geq \frac{storage}{obsfreq \times obssize}$ where $storage$ is the amount of storage on the monitor, $obsfreq$ is the maximum frequency with which observations are generated, and $obssize$ is the maximum size of an observation. We observe that the history length is inversely proportional to the size of observations, and therefore aggressive data reduction can extend the history length of a monitor.

The storage unit in a monitor receives data from the data reduction unit and stores it on some media. This storage is not necessarily local to the monitor as it could be a repository at some remote node. The storage unit may choose to overwrite or update previously stored observations according to some policy. One policy is first in-first out, where the oldest data is overwritten as new data becomes available. Another policy could prioritize the observations by type of traffic, so that traffic more likely to contain an intrusion attempt would be held longer in storage than benign traffic. IDIP [SDS00] includes a priority mechanism for storing observations so that more threatening traffic may be tracked longer than more benign traffic.

A monitor's reporting unit informs remote entities of observations either by answering queries or raising alarms. The reporting unit receives commands through the control unit from an analysis program to report on observations from storage. In the case of a query, the analysis program makes a request for observations matching certain criteria and the resulting observations are returned from storage. IDIP [SDS00] includes intrusion detection systems in its monitoring components. The intrusion detection systems can configure monitors so that when observations match certain patterns, the observations are reported to an analysis system.

The types of queries that a reporting unit can answer are dependent on the data reductions made before storage. Thumbprinting and SPIE use data reduction techniques similar to hash functions. Because of this, the data collected in these systems is only useful for reporting whether an observation of a piece of traffic with a certain hash value was observed.

The types of queries that a reporting unit can answer are also dependent on the amount of storage available. Some monitors such as the routers in DoSTracker [CD97] have no or little persistent storage for observations and are therefore limited to reporting observations that are currently being made by the monitor.

5 Analysis Programs

Passive NOISs have analysis programs that analyze observations from monitors to determine the origin of NDEs. These analysis programs run on one or more network components to determine information about the path or origin of an NDE by analyzing the observations reported by network monitors.

Analysis programs interact with network monitors by sending them commands via the network or other channels. Monitors may then respond to these commands using the same or alternate channels. The past work in passive NOI use the network to communicate with its monitors and therefore make a critical assumption. The assumption is that the network will not drop commands or responses between network monitors and the analysis program.

In the past work in which analysis programs have been described, the analysis program runs on nodes distinct from the destination of the NDE. This is a good choice, as it is likely the job of a network administrator to begin the analysis program. In some cases, a network intrusion detection system initiates an analysis program and therefore the analysis program may begin before the NDE has been received by its destination. Additionally, the observer could be administered by a third party such as an Internet service provider or law enforcement.

5.1 Types of Correlations

Analysis programs in passive NOISs use two types of correlation to process observations from monitors. The first is *node correlation* of which *origin correlation* is a special case. The second

type of correlation attempts to determine relationships between observations of NDEs and is called *NDE correlation.*

Correlation is a long-studied subject in statistics [Hac75]. In this reference model, we use correlation as a term for the process of relating certain types of entities in the model based on our observations. Mansfield et.al. [MOT+99] used traditional statistical correlation techniques to trace denial of service attacks, but others [BS02, MOT+99, BDKS00] have used the term correlation more loosely to describe determining relationships between NDEs and their origins and ancestors. We will use the term correlation with the understanding that we are not referring strictly to conventional statistical methods.

5.1.1 Node Correlation

In some cases, an analysis program may be able to determine a node in the path of an observed NDE directly from the observation. An example of node correlation is used in conjunction with Staniford's thumbprinting work [SC95]. When two extended connections are matched, the NDEs in the connection can be correlated with their upstream host in the path by their source addresses. The source address is probably correct for this link in an extended connection, as it is difficult to spoof the source address in an ongoing TCP stream such as those for which thumbprinting is effective.

Origin correlation is a special case of node correlation that determines identifying information about the origin directly from an observation of an NDE. Origin correlation has been used in traffic analysis of radio signals [oA48, Kah67] to determine the origin of radio transmissions.

None of the past work in passive origin identification in computer networks has focused on directly correlating NDEs to their origin, but have done so implicitly by making assumptions about the behavior of the network. For example, passive systems for tracing packets [CD97, MOT+99] have assumed that packets being traced were from hosts at the edge of the network and therefore, when a packet was found to have come from an edge router for an autonomous system, it was assumed that the packet originated in that autonomous system.

Other related work that could be useful for origin correlation is passive operating system fingerprinting [Pro02]. This work analyzes network traffic for behavior indicating that it was generated by a particular operating system. One example of this is the default value of the time to live field in IP packets. Some operating systems set it to 255 while others use lesser values. Fingerprinting of operating systems is not likely to be of great use, as it can only tell us which operating system and version thereof was running. Furthermore, the characteristics used by an operating system could be faked by a clever attacker.

A method that we have considered is based on trusting the network to properly decrement the IP time to live field. As its initial value can be no larger than 255, we know that the originator of an IP packet can be no more than $255 - ttl$ hops away from the observer. Given an accurate routing map for that time period, the set of possible originators could be reduced considerably. In response, an originator could choose an initial *ttl* so that it is 0 on arrival, thereby yielding no information about the origin. Furthermore, broken or compromised nodes on the path could modify the *ttl* arbitrarily.

5.1.2 NDE Correlation

Analysis programs in passive origin identification for computer networks have used a variety of NDE correlation techniques. NDE correlation refers to techniques that determine relationships between observations of NDEs so that the analysis system can form a chain of evidence that leads to a set of possible origins for an NDE.

Three types of NDE correlation have been used for passive origin identification. We call them shared origin correlation, cause correlation, and ancestor correlation.

Shared origin correlation (SOC) is the most general type of these correlations. Given an observation of an NDE output from a relay, $\omega' =$

$obs(m')$, each correlation type attempts to find observation(s) of NDEs input to the same relay, $\omega = obs(m)$, such that $origin(m) = origin(m')$.

The SOC approach makes assumptions about the protocol in use or type of attack being traced to make this correlation. In DoSTracker, a signature is created by the analysis agent of the NDE to be traced. Each router along the path in turn is queried for inputs matching the signature. When a match is found, it is not an ancestor of the original. The assumption is that because the packets match the signature, they are part of a group of NDEs that are generated by the same origin.

SOC is useful when monitors have small histories. Because the correlation is based on NDEs matching a criterion that indicates that they are likely to share the same origin, the monitor can wait to observe another NDE that matches the criterion instead of storing previously observed traffic.

The use of SOC can be a weakness as an attacker could send each NDE from a different origin. If this is the case, then a trace may fail as the next monitor may never see another NDE matching the signature.

For a relay, N, cause correlation determines one or more input observations, $\omega = obs(m)$, given an output observation, $\omega' = obs(m')$ such that $N.T(m, m')$. This is what an internal monitor may do directly, but to do so for an externally observed relay is more difficult. There may be many different input NDEs to N that produce a given output. Hence, there may be many NDEs observed as input that could have produced the output that we wish to cause-correlate.

A relay may also have state that is unknown to the analysis program, and transformations based on this state may make cause correlation difficult. An example is a relay such as a mix that encrypts its inputs with a strong secret key encryption system. There may be only one input that can be encrypted to form a given output, but it is computationally expensive to determine the matching input.

Ancestor correlation is a process that given two observations, determines if one is an ancestor of the other. This is a generalization of cause correlation because the cause of an NDE is an ancestor of the NDE as well. In some cases, ancestor correlation may actually be easier than cause correlation. If the route of an NDE crosses a relay multiple times, then determining all of these may be easier than picking one out as the exact cause of the NDE.

6 Sufficient Functionality For Passive Origin Identification

In this section, we introduce five mutually sufficient conditions for passive origin identification. They are network separation by trusted monitors, enough storage per monitor to accommodate the analysis, an analysis program to collect and process observations from the monitors, a trusted communication path between analysis programs and the monitors, and correlation of an input to a given output across every relay. Together, they form sufficient functionality for passive origin identification.

6.1 Network Separation by Trusted Monitors

If we can make no assumptions about the route taken by a piece of traffic other than best effort delivery based on destination address, we can make assertions about the origin identity based on observations of the network. One way is to separate the network into two or more components such that all NDEs flowing between the components are monitored. This is what we mean by network separation. It could also be the case that some network monitors are compromised so that they return false data or refuse to return data altogether. A trusted monitor will report data that reflects the data it observes when requested.

We have already seen network separation in "edge observed" networks and observed relays. Network separation is actually necessary for origin identification using passive techniques. To see why, consider a network with monitors inside such that no separation occurs. Hence, if

the monitored edges were removed, the network would remain connected. No nodes can be internally monitored in this scenario, as an internally monitored node would be separated from the network into a component of size one. This means that only edges are observed and that NDEs observed on an edge could come from any node in the network because there is always another path not observed. If there were not, the monitors would form a separation.

Additionally, we argue that separation is a sufficient placement for some level of passive origin identification. If we have monitors such that they separate the network into two components, then the route of the NDE either crosses the separator or it does not. If it does not, then the origin must be in the same component as the destination. If it does cross the separator multiple times, then the origin must be in the component from which the first observation was seen to arrive. If it was not, then there would be an observation of it crossing the separator at an earlier time.

From this argument, we can see that separation of the graph by monitored edges is a necessary and sufficient placement to achieve some level of origin identification, but this is deceptive. For instance, the separation might be a single internally monitored node. We then have an edge-monitored graph with one node representing the monitored node and its neighbor(s) representing the rest of the network. Certainly, observations at this node may tell us which of the neighboring subgraphs contains the origin, but the worst case is that the node has one neighbor representing all other nodes in the network. Hence, the number of candidates can be reduced to one in the unlikely event that the monitored node is the origin or $|V|-1$ otherwise. It is worst case in the sense that if the NDE originates in the remainder of the network, we can only determine that the flow originated somewhere in that remainder.

If a graph is separated by monitored edges, but one of the monitors is compromised, then an attacker could (among other things), falsify observations to cause a traceback algorithm to go towards the wrong component or fail to report observations to cause the search to stop earlier than if the monitor was uncompromised.

6.2 Storage

A monitor must have at least enough storage so that its history length exceeds the interval between observing an NDE and an eventual query for the observation. As discussed earlier, the needed history length is dependent on the time between observation of an NDE and its arrival at its destination. Additionally, the needed history length can be dependent on the search algorithm used by the analysis program [Shi99]. This dependence makes it difficult to state anything definitive about the history length without knowledge of the approach of the analysis program and the maximum latency of the network, as these may be of arbitrary or unbounded length.

6.3 The Rest

We now describe how the rest of the conditions can be combined to determine the origin of NDEs in a network based on the conditions already discussed. Because we can do so, these conditions are sufficient for passive origin identification. Furthermore, because these conditions are based on the components of the reference model, the reference model is useful for reasoning about origin concealment.

6.3.1 Internal Monitors

Assume that we have a network, $N = (V, E, IM, XM)$ where $XM = \emptyset$ and $IM = V$. N is an "edge observed" network, and because nodes are internally monitored, the relation between inputs and outputs may be observed. We now describe the internal directed search algorithm (commonly called traceback) in terms of this network based on the conditions described so far.

The NDE, m, arrives at v_k, and the administrator desires to trace it. We describe the steps of the algorithm for internal observers below.

1. An observation, $obs(m_k)$, is given to the analysis program as an argument.

2. In our analysis, we have assumed that a neighbor node, v_{k-1}, is apparent from the observation, and therefore the monitor at v_{k-1} is queried for an input that correlates with $obs(m_k)$. If the neighbor node is unknown from the observation, all neighbors can be queried this first time.
3. The result of the query is either an $obs(m_{k-1})$ such that $v_{k-1}.T(m_{k-1}, m_k)$ or v_{k-1} generated m_k. If it was generated there, then the algorithm halts reporting v_{k-1}.
4. The analysis program repeats the above with $obs(m_{k-1})$ as its argument.

In steps 1 and 5 we see that the analysis program coordinates the directed search. In step 2, the analysis program may query the monitor for either a predetermined pattern allowing the monitor to do the correlation or query the monitor for its entire storage, which allows the analysis program to do the query itself. Accurate NDE correlation allows the analysis program to make the correct choice for the next node to be queried.

As the analysis program repeats, it may query monitors that are increasingly distant. This process cannot proceed unless there is a communication path that is trustworthy between the analysis program and the monitors. If a monitor is untrustworthy, it could claim no correlations exist and the analysis would halt. If the monitors are trustworthy and the communication path does not modify or drop queries or responses, then the analysis program will get the correct next step in the path in each round of these steps. The result is that there will be only one node of the graph returned at the end of the run of the analysis program. Hence, we achieve origin identification.

6.3.2 External Monitors

A network with external monitors, $N = (V, E, IM = \emptyset, XM)$, can be reduced to an "edge observed" graph, $N' = (V', E', IM' = \emptyset, XM' = E')$ by the algorithm described earlier.

We can form an analysis program as we did for internal directed search. The difference is that this time we treat each node in N' as an observed relay with input and output observers composed from the actual monitors in XM. We begin in the node of the observed graph that contains v_k. The algorithm queries the input observer of the current node to find $v'_{k-1} \in V'$. This is done by cause-correlating the received observation with the flows observed input to the node. The prime indicates that the actual v_{k-1} may be grouped with other actual nodes in the "edge observed" graph. If the correlation step succeeds, a new node in V' is searched. If not, then then the algorithm replies with the nodes in V corresponding to the current node in V'.

Note that the difference between the internal and external algorithm is in the query step. In the internal case, one query needs to be made to an internal observer and the correlation is made. In the external case, the input observer is an abstraction of one or more actual monitors. The result is that many actual monitors may need to be queried to accomplish this step. Furthermore, if the correlation relies on timing or ordering of observations from different monitors, we have the problem of synchronizing distributed clocks [Lam78].

7 Forensic Implications

In this Section, we consider various forensics implications of the reference model based on the state of passive origin identification systems and features of the reference model. The first of our conclusions regarding forensic use of origin identification systems is that current techniques are only useful for investigation. The second is that current NOISs utility for investigation is limited by their dedication to tracing single types of traffic. These motivate certain aspects of our future work in this area.

Several issues come together to limit the utility of current NOI schemes for forensics other than during initial investigation phases. Most network-based schemes focus tightly on the issue of data reduction which is an important area as a short history length severely limits the utility of a NOIS. However, data reduction by definition eliminates redundant detail that could be useful

for corroborating certain items of evidence. Furthermore, with the exception of Carrier's work [Car01], tamper resistance has not been a goal of most schemes. Another issue with most NOISs is that as a trace approaches the origin, one can argue that its reliability and integrity decline. Consider that the NOIS is relying on data collected nearer an attacker and therefore more likely to be under his control. This may be a result of their emphasis on research or an attempt to minimize the computational and storage requirements of the system. Hence, most passive systems have nothing to protect the integrity of the evidence and this seems somewhat at odds with other design goals of the systems themselves. Future research work should go into addressing these requirements.

Another major forensics problem facing current NOISs is that they focus on single types of transformations or are host-based. Most passive NOIS research that is not host-based only addresses a single type of traffic transformation e.g. island hopping, IP spoofing, etc. Hence, tracing complex attacks will only work as far back as that single type of transformation is used. Then it is up to the investigator to find the previous step. Host-based solutions are valuable, but are difficult to use between organizations as there are definitely privacy concerns involved. Further, analysis of host-based results may require human intervention. This intervention consumes valuable time that may exceed history lengths of upstream NOISs. Hence, we suggest that future NOISs must take multiple types of transformations into account and work to integrate different approaches including host-based and network-based systems and active methods as well. This is especially true as more advanced distributed attack tools utilize modern anonymity techniques and more layers of indirection.

8 Conclusions

We have presented a reference model of passive network origin identification. It is composed of a functional model of a network monitor that communicates with an analysis program. The network monitor has a control unit for interpreting commands from an analysis program and a reporting unit that provides observation information to the analysis program. The monitor also has functional units for selecting, reducing in size, and storing observations of NDEs and the behavior of relays. The analysis program can query the monitors for observations and correlate observations to determine the origin of NDEs.

We have described how several conditions are sufficient for determining the origin of NDEs. Because these conditions are derived from the reference model discussed in this chapter, the model has therefore been useful for reasoning about the observation of origin identity. Additionally, we have reasoned about the relation between storage and history length of monitors.

Finally, we have considered some conflicting issues and areas of future work for network forensics. It is our hope that this general study of origin identification will lead to better understanding and communication between researchers and practitioners.

References

[BDKS00] Florian Buchholz, Thomas E. Daniels, Benjamin A. Kuperman, and Clay Shields. Packet tracker final report. Technical Report CERIAS TR 2000-23, Center for Education and Research in Information Security and Assurance (CERIAS), Purdue University, West Lafayette, Indiana, 2000. Available at `https://www.cerias.purdue.edu/papers/archive/2000-24.pdf`.

[Bel00] Steven M. Bellovin. Internet draft: ICMP traceback messages. Available at `http://www.ietf.org/internet-drafts/draft-bellovin-itrace-00.txt`, March 2000.

[BS02] Florian P. Buchholz and Clay Shields. Providing process origin information to aid in network

traceback. In *Proceedings of the Annual USENIX Technical Conference*, Monterey, California, 2002.

[Car01] Brian Carrier. A recursive TCP session token protocol for use in computer forensics and traceback. Master's thesis, Purdue University, West Lafayette, Indiana, May 2001.

[CD97] H. Chang and D.Drew. DoS-Tracker. This was a publically available PERL script that attempted to trace a denial-of-service attack through a series of Cisco routers. It was released into the public domain, but later withdrawn. Copies are still available on some websites., June 1997.

[Hac75] Ian Hacking. *The emergence of probability: a philosophical study of early ideas about probability, induction and statistical inference*. Cambridge University Press, London, United Kingdom, 1975.

[Kah67] David Kahn. *The Codebreakers: The Story of Secret Writing*. Scribner, 1967.

[Lam78] Leslie Lamport. Time, clocks and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.

[MOT+99] G. Mansfield, K. Ohta, Y. Takei, N. Kato, and Y. Nemoto. Towards Trapping Wily Intruders in the Large. In *Proceedings of the Second Annual Workshop in Recent Advances in Intrusion Detection(RAID)*, West Lafayette, Indiana, September 1999.

[oA48] Department of Army. *Fundamentals of Traffic Analysis (Radio Telegraph)*. Aegean Park Press, Laguna Hills, California, 1948. Also known as Department of the Army Technical Manual TM 32-250 and Department of the Air Force Manual AFM 100-80.

[Pro02] The Honeynet Project. Know your enemy: Passive fingerprinting. Technical Report Available at `http://project.honeynet.org/papers/finger/`, The Honeynet Project, March 2002.

[Row99] Jeff Rowe. Intrusion detection and isolation protocol: Automated response to attacks. Presentation at RAID'99, September 1999.

[SC95] S. Staniford-Chen. Distributed tracing of intruders. Master's thesis, University of California at Davis, 1995.

[SCH95] S. Staniford-Chen and L.T. Heberlein. Holding intruders accountable on the Internet. In *Proc. of the 1995 IEEE Symposium on Security and Privacy*, pages 39–49, Oakland, California, May 1995.

[SDS00] Dan Schnackenberg, Kelly Djahandari, and Dan Sterne. Infrastructure for intrusion detection and response. In *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX) 2000*, Hilton Head, South Carolina, January 2000. IEEE.

[Shi99] C. Shields. *Secure Hierarchical Multicast Routing and Multicast Internet Anonymity*. PhD thesis, University of California, Santa Cruz, June 1999.

[SPS+01] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Stephen T. Ken, and W. Timothy Strayer. Hash-based IP traceback. In *Proceeding of the ACM*

SIGCOMM'01, San Diego, California, August 2001.

[SWKA00a] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical network support for ip traceback. In *Proceedings of the 2000 ACM SIGCOMM Conference*, pages 295–306, Stockholm, Sweden, August 2000.

[SWKA00b] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical network support for ip traceback. Technical Report UW-CSE-2000-02-01, University of Washington, 2000.

[Tan88] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, Inc., second edition, 1988.