

ForNet: A Distributed Forensics Network

Kulesh Shanmugasundaram, Nasir Memon, Anubhav Savant, and Herve Bronnimann

{kulesh, anubhav}@isis.poly.edu, {memon, hbr}@poly.edu
Department of Computer and Information Science
Polytechnic University
Brooklyn, NY 11201
USA

Abstract. This paper introduces *ForNet*, a distributed network logging mechanism to aid digital forensics over wide area networks. We describe the need for such a system, review related work, present the architecture of the system, and discuss key research issues.

1 Introduction

Computer networks have become ubiquitous and an integral part of a nation's critical infrastructure. For example, the Internet is now regarded as an economic platform and a vehicle for information dissemination at an unprecedented scale to the worlds population. However, the last few years have taught us that these very networks are vulnerable to attacks and misuse. Hence, mitigating threats to networks have become one of the most important tasks of several government and private entities. Recent attacks on critical network infrastructures are evidence that defensive mechanisms, such as firewalls and intrusion detection systems, alone are not enough to protect a network. Lack of attack attribution mechanisms on our networks provides an excellent cloak to perpetrators to remain anonymous before, during, and after their attacks. Most often we are not only unable to prevent the attacks but also unable to identify the source of the attacks. In order to guarantee the security and the survival of future networks we need to complement the defensive mechanisms with additional capabilities to track-and-trace attacks on wide area networks.

The problem with most defensive measures is that they are fail-open by design and hence when they fail the attackers have the opportunity to leave the crime scene without any evidence. In order to track down the perpetrators, forensic analysts currently rely on manually examining packet-logs and host-logs to reconstruct the events that led to an attack. This process has the following drawbacks:

- **Large Volume of Data:** Growth of network traffic out-paces Moore's law [41] making prolonged storage, processing, and sharing of raw network data infeasible. Most of the approaches to evidence collection have focused on improving the performance of packet-loggers to keep up with ever increasing network speeds and have failed to exploit the inherent hierarchy of networks or the availability of various synopsis techniques to collect evidence efficiently.

- **Incompleteness of Logs:** Most network administrators rely on alerts from intrusion detection systems and the resulting packet-logs for forensic analysis. It is important to note that intrusion detection systems log packets only when an alert is triggered. Hence, a new attack or an attack currently not defined by the security policy is not recorded by the system. Some networks are equipped with packet-loggers which log packets indiscriminately. However, due to cost considerations and associated storage requirements, such packet loggers are usually deployed at network edges and do not witness network events, such as insider attacks, that never cross network boundaries.
- **Long Response Times:** Most of the investigative process is manual. Since attacks often span multiple administrative domains, this makes response times undesirably long. Since digital evidence is malleable it is important to reduce response times so that evidence can be secured on time.
- **Lack of Mechanisms to Share Logs:** The inability of existing forensic mechanisms to share evidence and lack of mechanisms to query networks prohibit forensic analysts from exploring networks incrementally while tracking and tracing an attack. This in turn results in inefficient allocation of resources because the analysts are unable to confine their search for perpetrators to a particular network.
- **Unreliable Logging Mechanisms:** Logging mechanisms on hosts are not reliable and can easily be circumvented by adversaries. Increasing use of wireless networks and support for mobility enable hosts to join a network from any arbitrary point and makes enforcement of prudent host logging policies difficult.

As more and more failures of defensive mechanisms result in financial damages and cause significant threats to our critical infrastructure, there will be an increased need for attack attribution so that criminal acts do not go unpunished. In this context, digital forensics will play an increasingly vital role in the security of future networks. In fact, we envision that future networks will be equipped with forensic components that complement existing defensive mechanisms to provide viable deterrence to malicious activity by increasing our ability to track and trace attacks to their source.

In this paper we present the design of a network forensics system, *ForNet*, that aims to address the lack of effective tools for aiding investigation of malicious activity on the Internet. ForNet is a network of two functional components. A Synopsis Appliance, called *SynApp*, is designed to summarize and remember network events in its vicinity for a prolonged period of time and be able to attest to these events with certain level of confidence. A *Forensic Server* is a centralized authority for a domain that manages a set of SynApps in that domain. A Forensic Server receives queries from outside its domain, processes them in co-operation with the SynApps within its domain and returns query results back to the senders after authentication and certification.

One key idea behind our network forensics system is that it builds and stores summaries of network events, based on which queries are answered. Ideally, a network forensics system should provide complete and accurate information about network events. However, this is not feasible in practice due to storage limitations. It is our contention that even if a network forensics system is unable to provide complete and accurate information about network events, as long as we have intelligent summaries, snippets

of approximate information can be put together to form the “big picture,” much like corroborating evidence in classical forensics.

For example, let us consider the latest Internet worm outbreak, SQL Slammer Worm, as a possible investigation scenario where an analyst has to find the origin of the worm. The worm spreads by infecting unpatched SQL Servers running on UDP port 1434. Assuming ForNet is widely deployed on the Internet the analyst needs to determine from which region of the Internet the worm began its propagation. Since SynApps keep track of various events in their local environment, the analyst will be able to determine a spike in traffic to port 1434 on any network. Starting from an arbitrary network the analyst can now query the network about its first observation in increased activity to port 1434 and recursively query any network which reports the earliest time. These recursive queries will eventually lead us to a particular network from which the worm started its propagation. Now the analyst can focus their investigative resources into a particular network to locate a host that sent out the first malignant packet to port 1434. This would help them associate a particular host to the outbreak. When further investigation on the host turns up new leads, ForNet can be used in a similar fashion to find the perpetrator who actually unleashed the worm.

Currently there is no infrastructure to provide valuable, trustworthy information about network events. We believe ForNet will fill this gap. The rest of this paper is organized as follows: the next section presents a brief overview of related work. Section 3 presents design goals of ForNet and an overview of the system. In Section 4 we discuss synopsis techniques, followed by a discussion of security issues in the design of ForNet in Section 5 and we conclude in Section 6 with a discussion on future work.

2 Related Work

Network Forensics: There has been very little work done in network forensics. Mnemosyne is a dynamically configurable advanced packet capture application that supports multi-stream capturing, sliding-window based logging to conserve space, and query support on collected packets [29]. In [42], a general framework in terms of enterprise network and computer related policies to facilitate investigation is presented. Also, in the recent past, researchers have proposed some clever solutions to the Denial Of Service (DoS) problem by tracing IP packets back to their source (IP traceback) [6, 11, 13, 26, 35, 38, 39]. Most of this work can be grouped into two main categories: one in which no extra network packets are generated [11, 13, 35, 38, 39] and the other in which a few extra network packets are generated [6, 26]. The former is either based on probabilistic packet marking which overloads existing header fields (e.g. IP fragment ID field) to succinctly encode the path traversed by a packet in a manner that will have minimal impact on existing users or based on keeping the digest of all the IP packets in the infrastructure itself (e.g. routers). Thus, given an input IP packet, it can be reliably traced back to its origin. In the latter technique, a router picks a network packet with a small probability (e.g. 1 in 20,000) and sends a traceback packet as an ICMP message to the destination of the packet with the router’s own IP as the source. During a denial of service attack the victim will receive sufficient traceback packets to reconstruct the attack path. Recently,

another system to track malicious emails (in fact any email) has been proposed, which aims to reduce the spread of self-replicating viruses [7].

Intrusion Detection Systems: Intrusion Detection Systems (IDS) have been studied extensively over the past few years and the main body of work can be categorized into two groups, signature based and anomaly detection systems [14]. Signature-based methods rely on a specific signature of an intrusion which when found triggers an alarm [23, 25]. Such systems cannot detect novel attacks as evident by recent Internet worm activities. On the other hand, anomaly detection systems rely on characterizing normal operations of a network or a host and attempt to detect significant deviations from the norm to trigger an alarm [18, 24, 31, 32]. Although anomaly detection systems can detect novel attacks to a certain extent they are not a viable solution because of the unacceptable rate of false alarms [2]. Hybrid solutions [1], that combine signature based systems and anomaly detection systems, have been suggested to decrease false alarm rates and are used in commercial products. SHADOW and EMERALD are distributed IDS capable of collecting and processing data from various points on a network. As far as we know these are the earliest schemes to collect data in a distributed manner for security related analysis. In these systems raw data is collected and transferred to a central collection facility for analysis. IDS' are not always suitable for forensic analysis as they only catch the known or the unusual. But crimes are often committed by insiders using new and unknown methods. It is conceivable that an IDS and a forensics system can co-operate with each other, but this is beyond the scope of this paper. We envisage the community moving in this direction, once we have demonstrated a working prototype.

Synopsis Techniques for Managing Data Streams: Data Stream Management Systems (DSMS) have been proposed [5] to integrate data collection and processing of network streams in order to support on-line processing for various network management applications. Data stream management systems work on continuous streams of data with stringent bounds on space and processing power to produce best estimate results [3, 30]. With such volumes of data, there is no time to do detailed analysis but only some synopses of the data streams can be extracted and archived. There are various approaches to building synopses of data streams [19] (histograms [21, 27, 28, 40], samples [4, 19], wavelets [20], decision trees [15, 22], sliding windows [4, 12], Bloom-filters [8, 9]) and each is adapted to a certain kind of queries. Often, the kind of aggregate queries they answer are not suitable for forensics, as they tend to focus on the recent past (sliding windows), or on global aggregate quantities (histograms, wavelets), instead of focusing on precise periods or events in the past.

Industry Efforts: It is only until recently that commercial organizations have realized the significance of a forensics system in aiding investigation for the cause of network security problems. In fact, at the time of writing Network Associates introduced a forensics tool called InfiniStream [34], which is a brute force approach that will not scale to a network of appliances. Also, there does not seem to be any concept of a network of Forensics Servers that we describe. Another product along similar lines is from Sandstorm Enterprise called NetIntercept [16].

3 Overview of ForNet

In this section we provide an overview of ForNet, and describe the architecture of one of its components, the SynApp. ForNet is a distributed network logging mechanism to aid digital forensics over wide area networks. It is highly scalable and hence can keep up with ever increasing volume of data that goes through our networks. Unlike traditional packet-loggers, ForNet uses synopsis techniques to represent enormous volume of network traffic in a space-efficient manner so that it can be stored, processed, or transported across networks to answer queries.

3.1 ForNet Blueprint

ForNet is made of two main components: SynApps and Forensic Servers. SynApp is an appliance that can be integrated into networking components, such as switches and routers, that can summarize and remember network events in its vicinity for a prolonged period of time and be able to attest to these events with certain level of confidence. Although a single SynApp functioning on its own can provide very useful information to a forensic analyst, a network of such co-operating appliances (even across a corporate intranet) would bring powerful new possibilities to the types of information that could be inferred from the combined synopses. Networking SynApps would also help them share data, storage, and let them collaborate in answering queries accurately. These SynApps can be arranged in a pure peer-to-peer architecture to collaborate with each other in the absence of centralized control. A hierarchical architecture is simpler (See Fig. 1) and would also work better with the given structure of the Internet.

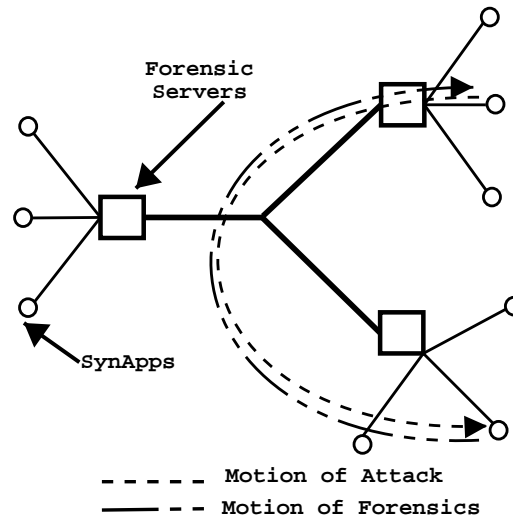


Fig. 1. Hierarchical Architecture of ForNet

In the hierarchical architecture all the SynApps within a domain form a network and are associated with the Forensic Server of that domain. Like the DNS servers of today, we envision future networks to have Forensic Servers as a critical component of their infrastructure. Forensic Server functions as a centralized administrative control for the domain, receiving queries from outside its domain to pass them on to the query processor and storage management unit (which could be located in the SynApp or in the Forensic Server) after authentication and sending query results back after certification. Network of SynApps form the first level of hierarchy in the hierarchical architecture of ForNet. Naturally, Forensic Servers themselves can be networked for inter-domain collaboration which forms the second level of the hierarchy. Queries that need to cross domain boundaries go through appropriate Forensic Servers. A Forensic Server is the only gateway to queries sent to a domain from outside the domain boundaries. A query sent to a domain is addressed to the Forensic Server of the domain, authenticated by the server and passed on to appropriate SynApps in the domain. Likewise, results from the SynApps are sent to the Forensic Server that is in control of the domain where it is certified and sent back. In practice queries would begin from the leaf nodes from a branch in the hierarchy, traverse Forensic Servers in higher levels, and end up in leaf nodes in another branch. Queries would usually travel in the opposite direction of the attack or crime. In the beginning queries will be more general as the analyst tries to pin-point the origin of the attack and then the queries become very precise as she close in on the source of the attack.

3.2 Architecture of a SynApp

Here we show how we intend to package various synopsis techniques discussed in the next section into a single appliance, the SynApp, which we have begun to realize first in software, and then in its final form as a small network appliance dedicated to synopsis-ing network traffic. The appliance has two abstract components, a network resident *synopsis engine* and the *Forensics Server* - a possibly external query processing and storage management unit. These are further divided into several functional modules. Fig. 2 illustrates the overall architecture of the appliance with an integrated Forensics Server. We envision that such an appliance can be seamlessly integrated into networking components or it can be attached to the network right after a router.

The modular architecture allows for a variety of configurations of SynApps and Forensic Server in a network (to satisfy budgetary and security policy requirements of an organization). Cheaper SynApps can be designed without persistent storage and query processor. In this case, SynApps periodically send synopses to the Forensic Server which will handle storage and queries itself. On the other hand, SynApps can be independent of Forensic Server. In this setup a SynApp overwrites its old storage whenever it needs more memory and functions as a self-contained unit sparing the expenses of maintaining a Forensic Server.

We briefly describe the functionality of each element below:

- Network filter: The appliance may not intend to process every single packet the router sees and uses the filter to extract useful packets.

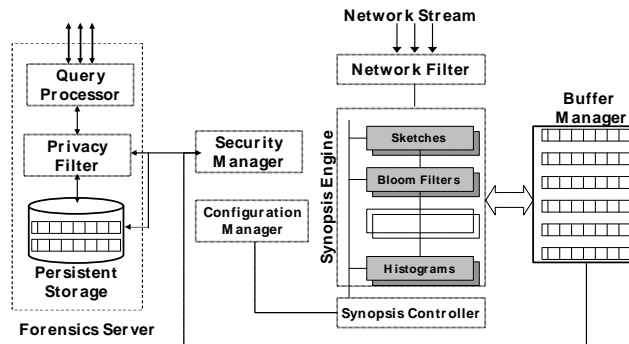


Fig. 2. Architecture of a SynApp with an Integrated Forensics Server

- Synopsis Engine: The engine contains data structures and algorithms that can be used to represent network traffics succinctly. The data structures and algorithms can be tuned (for example via a trade-off between space-efficiency and accuracy) via the configuration manager. Each packet that passes through the network filter is pipelined through the synopsis techniques implemented in the synopsis engine and each technique decides to process or not to process a packet based on the signal it receives from the Synopsis Controller.
- Synopsis Controller: The Synopsis controller instructs synopsis techniques in the engine whether to process or not to process a given packet. A rule-set written by the user and processed by the Configuration Manager allows the user to modify how packets should be handled by the synopsis engine. Separating the design of data structures and algorithms (in the synopsis engine) and the function of these techniques (how they are mixed and matched, in the synopsis controller) results in an extensible architecture which can be programmed to record various events.
- Configuration Manager: Configuration manager is the interface between a network administrator and the appliance which allows the administrator to fine-tune various operations of the appliance.
- Security Manager: Each query that will be served by the query processor must be authenticated by the security manager so that rogue queries from unknown users are simply ignored. In addition, security manager signs and time-stamps each data item written to the database to protect integrity and to be able to use in a court of law.
- Storage Management and Query Processor: Query processor handles all database access and query processing issues and is discussed in more detail below when we discuss the Forensics Server.

3.3 Design Considerations:

Experiences we have had with a simple proof-of-concept ForNet network in a large network have steered us toward achieving the following design goals for a forensically sound network logging mechanism:

Capture of Complete & Correct Evidence: Network speeds have grown considerably over the years making it difficult to capture raw network traffic in its entirety. Therefore, SynApps should be able to capture network traffic and create appropriate synopses at line speeds. Our goal is to implement SynApps in hardware by using network processors such that synopses can be created at line speed. Traditional packet-loggers encounter severe bottlenecks when transferring raw network data from high-speed network media to slower storage media. SynApps get around this bottleneck by transferring synopsis of network traffic which is only a fraction of the size of actual traffic. Furthermore, synopsis techniques also allow us to quantify the errors during transformation of raw data such that an analyst can quantify the correctness of evidence upon retrieval.

Longevity & Accessibility of Evidence: Captured network data must be stored for a prolonged period of time as we don't know when we might need the data. Synopses allow us to archive evidence for a longer time (compared to storing raw network data) as they represent large amount of network data succinctly. Prompt and secure access to evidence is important to solving a crime on time. SynApps support secure remote access to evidence via a query mechanism.

Ubiquity of SynApps: In order for SynApps to capture evidence they must be omnipresent on a network. Hardware design of SynApps will be such that they can be seamlessly integrated into networking components and their presence on the network can be as ubiquitous as that of networking components.

Security & Privacy: It is safer to assume various components of ForNet will be potential targets of attacks themselves. We discuss some attacks on ForNet in Section 5. Moreover, any forensic tool that monitors networks must consider privacy implications and support adequate compartmentalization of data they store.

Modular & Scalable Design: Modular design of various components of ForNet allows for new synopsis techniques to be introduced into the system without expensive re-designs. In order to keep up with ever increasing network traffic ForNet must scale well as a whole. Therefore, various components of ForNet, from communication protocols to query processors, must be designed with scalability in mind.

4 Synopsis Techniques

Evidence of crimes can be found in network events that are well defined by various fields in packet headers and/or by application dependent payloads. A brute force approach that entails storing entire packets is not feasible for reasons discussed in Section 1. Therefore, we adopt synopsis techniques to record the network events in a succinct form for a prolonged period of time. A *synopsis* is a succinct representation of massive base data that can provide approximate answers to queries about base data within a predefined level of confidence. Defining a network event can be as simple as concatenating a set of values from fields in the header of a single packet or it can be as complex as extracting various features from a collection of several packets. For example, a TCP *connection establishment event* can be defined by a pair of IP addresses

and a pair of port numbers from a TCP packet with SYN flag set. On the other hand, *port scanning event* cannot be defined simply by a set of fields in the packet headers or by the payload. Defining a port scan would involve computing a variety of properties of network traffic, such as packet arrival rate per host and distribution of source IP addresses etc. In this section we briefly present a few useful synopsis techniques that can be used to capture network events succinctly. A detailed discussion of these and other techniques can be found in [37].

Connection Records: Network events, such as connection establishment and tear-down, are usually well defined by various fields in the packet headers. A simple scheme that is sufficient to answer queries of the form *who established a connection during time t ?* and *how long did the connection last?* is to store the pair of endpoints (host, port) with the TCP flags. We call this a *connection record*. We can store connection records for all TCP connections, within a given time window, if any of the three flags SYN, FIN, or RST is set. The cost for this is roughly 26 bytes per TCP connection. So for example, in a moderately loaded network, where on an average we expect to see 100 connections a second, this would require about 1300 bytes to be timestamped and stored every second. The total cost over a day of these records would only be about 200MB.

Bloom Filters: A different and potentially more space-efficient approach to track network events can be arrived at by using Bloom filters [8]. A Bloom filter is a probabilistic algorithm to quickly test membership in a large set using multiple hash functions into a single array of bits. Space efficiency is achieved at the cost of a small probability of false positives. For example, we adapt Bloom filters to keep track of connections by periodically inserting the hash of the pair of IP addresses, the port numbers and TCP flags of packets with TCP SYN, FIN, or RST flags set. To check whether a particular connection was seen during a time window, we compute the necessary hashes (on IP, port, and TCP flag fields) and test it against all available Bloom filters for the time window. If a Bloom filter answers “yes” then we know the connection packet was seen with a confidence level determined by the false positive rate of the filter. Otherwise we know for certain that the corresponding packet was not seen.

A Bloom filter with a false positive rate of 4.27×10^{-5} uses only 42 bits (or about 5 bytes) of storage per connection. In comparison, a connection record uses 26 bytes to store a connection. Note, however, to use the Bloom filters we need the information about connection headers and these headers can only be obtained where connection records are maintained. The inherent hierarchy of the network plays a vital role in solving this problem. In the ForNet hierarchy (See Fig 1) SynApps closer to the end hosts maintain connection records whereas SynApps closer to the network edge maintain the Bloom filters. With such an arrangement we can also identify spoofing as the connections can now be confirmed not only at the subnets but also closer to the core of the network. Besides, the space-efficiency of Bloom filters is suitable to represent numerous connections succinctly at network edges.

Another adaptation of a Bloom filter is to represent a histogram succinctly. A counting Bloom filter uses an array of counters instead of an array of bits to keep a count of distinct elements in a set. Instead of flipping a bit, counting Bloom filters increment the corresponding counter by one. The counting Bloom filter can be used to count various

properties of network traffic, like packet arrival rate per host per port. This information can be used to determine various patterns in network traffic.

Hierarchical Bloom Filters: Although connection records and Bloom filters are very useful in tracking and tracing network connections, they do not help in monitoring packet payloads. Monitoring and recording payload of network traffic is a challenging task for two reasons:

- *Lack of structure:* Packet headers have a pre-defined structure and semantics are interpreted uniformly across the network. The data payload on the other hand is application dependent and its interpretation varies across various applications.
- *Amount of data:* Packet headers are relatively small compared to payload hence compact storage of packet headers is feasible even at higher network speeds. The payload of a packet, on the other hand, can be several hundred bytes long and keeping up with the amount of data is a challenging task.

In order to monitor payloads, we propose an efficient method, called Hierarchical Bloom Filters, that supports limited wild-card queries on payloads and allows for space-accuracy trade-offs. Hierarchical Bloom filters can support wild card queries of the form “ $S_0S_1 * S_3$ ”. A detailed description of Hierarchical Bloom Filters is beyond the scope of this paper. A detailed description can be found in [36]

Sampling & Histograms: In addition to connection records and Bloom filters it is useful to sample the traffic to extract higher level features. High-level synopses provide trends which may be useful to *guide* the search while investigating a crime, which may later be confirmed with other synopsis techniques that remember all the connections and can recall with high probability if a packet was seen on the network. For instance, a multi-dimensional histogram will identify that there was a high proportion of SYN packets from two machines on various ports, which may indicate a port scan. High frequency of activity on a single port from various machines to a single target may indicate denial-of-service attack (DoS). It can also gather trends across time, for example detecting a slow scans. A sample may also indicate a time period for a certain connection provided this connection has enough packets so that at least one is included in the sample.

In general, high-level data is susceptible to data mining. Since the purpose is forensics and not intrusion detection, a later processing is acceptable as long as the mining has sufficient and representative data to use. A sample is adapted in this case because it keeps a representation of the data which can later be mined in unexpected ways whereas histograms and wavelet representation cannot.

5 Security of ForNet

The usefulness of information obtained from ForNet relies on the integrity, accuracy, and authenticity of results it generates. Furthermore, ForNet itself has to be resilient to attacks by an adversary. In this section we discuss some measures to provide integrity and authenticity of data on ForNet and possible attacks on ForNet itself.

5.1 Security Measures in ForNet

Security primitives required for data integrity and authentication are provided by the Security Manager. Integrity of evidence on ForNet is guaranteed by digitally signing time-stamped synopses along with necessary meta data, such as false positive rates and hash keys, before committing them to the database. Security manager also enforces access control to data by means of privacy policies. All queries should be signed by querier and security manager verifies access control privileges of the querier upon receiving the query. If the query doesn't violate any privacy policies then the query processor retrieves necessary data from storage. Upon successful completion of integrity checks on the data by security manager query processor is allowed to process the data. Finally, results generated by the query processor are certified at the Forensic Server and sent to the querier along with meta data required to quantify the accuracy of results.

5.2 Attacks on ForNet

As a passive network security tool ForNet is vulnerable to most of the attacks described in [31, 33]. In addition, being a storage unit for network traffic and processing queries from untrusted remote hosts introduce new challenges as well. A notable distinction on the effects of these attacks on intrusion detection systems and forensic systems, like ForNet, is that while in an IDS they may subvert the system to avoid recording of evidence, on ForNet such attacks would create problems during query processing and data interpretation but cannot subvert the system to avoid recording a set of packets. Since query processing is done offline and does not need the strict real time response of an IDS, such attacks can be countered more effectively in ForNet. Of course, we assume the adversary has full knowledge of the inner workings of various ForNet components, such as the data being stored at the Forensic Servers, algorithms used by SynApps and query processors, security primitives in place, and their use by security manager. We now describe possible attacks on various components of ForNet.

Forensic Server: Forensic Server needs to be protected very well, like the DNS servers, as it contains valuable network evidence. Data corruption on Forensic Server may lead to discarding results produced by the server. In order to minimize the damage data shall be recorded to a read-only medium frequently. In addition a rogue Forensic Server can exploit the trust relationships in the ForNet hierarchy and attack other servers by sending rogue queries. Appropriate resource allocation strategies shall be in place to rectify resource starvation attacks on Forensic Servers.

Query Processor: Since the Forensic Server processes queries from remote servers an adversary can send numerous queries to a server and consume its resources or on the other hand send a complicated query to bog down its resources.

Storage Unit: Needless to say a simple denial of service attack would be to send lot of data that will pass through the network filter and be processed and stored by the storage unit in order to fill-up the storage unit. This attack is especially effective when the SynApps are setup to overwrite current data or retire it periodically to secondary or

tertiary storage. An attacker can potentially flush useful data by sending in enormous amount of spurious traffic to a SynApps. Data retirement policies shall be designed with care to counter such attacks. For example, keeping a representative sample of the retired data shall alleviate the effects of such an attack.

Synopsis Engine: The synopsis engine uses various algorithms to represent network data succinctly. An adversary with the knowledge of how an algorithm works can trick the algorithm into recording lesser or no evidence at all. For example, an attacker can fool a sampling algorithm into sampling lesser evidence by “hiding” a few attack packets among a lot of bogus packets that are eventually discarded by the end-host but are considered valid by the sampling algorithm. An attacker can for example create lot of packets with small TTL such that they would never reach end-hosts, or packets with invalid headers that will be discarded by end-hosts, or simply increase the number of packets via IP fragmentation and hide the real attack packets among them. Creating such a cloak requires an attacker to generate numerous packets. An increase in packet arrival rate can easily be detected by SynApps. For example, a simple synopsis technique, like the counting Bloom filters, can keep track of the history of packet arrival rate per host. When the arrival rate exceeds a certain threshold the SynApps can sample the excess data knowing it may be part of an attack. Such a mechanism will of course has to be deployed at SynApps closers to hosts and not at the ones closer to the core of the network.

Forensic Analyst: In the previous case, the adversary manipulated synopsis algorithms’ interpretation of evidence but it is important to note that he can also manipulate how an analyst or the query processor interprets the evidence. For example, the adversary can send a FIN or RST packet to a host with which he has established a connection such that the host would take no action for the packet– for example FIN or RST with wrong sequence numbers– however connection records reflects a connection termination as it simply records any packet with TCP flags set. Although the connection records will have subsequent real connection termination packets if the analyst or the query processor only looks for the first FIN or RST packet following a SYN it would seem the connection is terminated earlier. Note that unlike the previous attack, in this attack data is available in ForNet however it is never processed as the query interpretation of connection time is first FIN or RST packet that follows the SYN packet. A persistent analyst will find this ambiguity in the connection records.

6 Conclusions and Future Work

In this paper we have motivated the need for a network forensics system that aids in the investigation of crimes committed on or via a network. We also present a high level description of the design of a distributed forensics network called ForNet which aims to fill this need. At the core of ForNet is a SynApp which creates synopses of network activity. Multiple such SynApps within a domain and connected to other domains via a Forensics Server form a hierarchical structure which comprises ForNet. We present an overview of the architecture of a SynApp and some simple examples by means which synopses of network events could be constructed.

Although we have described our vision and goals in this paper, a lot of effort still remains to bring our ideas to fruition. There are many important problems that need to be solved, which include:

- **Identification of Useful Network Events:** Cybercrimes are committed over networks and the network can be viewed as a virtual “crime scene” that holds critical evidence based on events before, during, and after a crime. The problem is that we do not know when, where, and how a crime will be committed beforehand. The challenge then is to identify useful network events, such as connection establishment, DNS queries, fragmentation of IP packets, etc., and choose a minimum representative set that would potentially be evidence in a variety of cybercrimes.
- **Developing Efficient Synopses:** There are two types of traffic on the Internet - packets belonging to a connection and connectionless traffic. The traffic corresponding to connection consists of three distinct phases when viewed at the network protocol level: connection establishment, data transfer and connection termination. In order to keep a synopsis of such a connection we may need to store all the “interesting events” that happened during its lifetime. The challenging issue here is to decide what information or events to store that would represent the connection as succinctly as possible and at the same time captures all relevant events. Clearly there is a trade-off between space and accuracy that needs to be made. The problem becomes more difficult when dealing with connectionless traffic as there is no binding glue of a connection that imposes some structure on the nature of the traffic. Here we basically have to decide which packets to keep a record of and what type of record? For example histograms can be used to keep track of frequencies of certain network traffic characteristics, like connections to a specific port within a time window. Also, each synopsis technique in itself may not be able to satisfy desired space-accuracy requirements. The challenge then is how do we cascade synopsis techniques working together to record certain network events? What combinations of techniques are suitable for recording various types of network events efficiently? What are the implications of this cascading on false positive or false negative analysis?
- **Integration of Information from Synopses Across Multiple Networks:** A single SynApp will only be able to answer queries about the events within its vicinity. While this could be of tremendous use by itself, the real power of a logging mechanism would be realized when multiple SynApp are networked together to answer queries about events in the larger network they span. For example, where as a single SynApp may say when a worm appeared in a given network, together they can potentially help locate the origin of the worm by finding out where it appeared first. This would then set us on the right track to finding the culprit who created the worm. So the benefit of networking the individual appliances is compelling, but its realization brings out new challenges. How do we connect them so that they can jointly respond to queries? How do we propagate queries to other (known) peers? Is there a minimal yet complete protocol that could facilitate the cooperation among synopses appliances? How could such a collaborative system provide guarantees about the privacy of the data handled?

- **Security of ForNet:** As we discussed in Section 5, various components of ForNet itself are targets for an attack by an adversary. Besides the attacks described in that section we need to explore further to identify attacks on various synopsis techniques and counter measures for these attacks. Finally, incorporate these counter measures into SynApps so that evidence gathered by them are still valid.

References

1. S. Axelsson. Research in intrusion-detection systems: A survey. Technical Report No 98-17, December 1998.
2. S. Axelsson. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *Proceedings of the ACM Conference on Computer and Communication Security*, November 1999.
3. B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Symposium on Principles of Database Systems*, Madison, Wisconsin, USA, June 2002. ACM SIGMOD.
4. B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *Proceedings of 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2002.
5. S. Babu, L. Subramanian, and J. Widom. A data stream management system for network traffic management. In *Workshop on Network-Related Data Management*, 2001.
6. S. M. Bellovin, M. Leech, and T. Taylor. ICMP traceback messages. In *Internet Draft draft-ietf-itrace-01.txt (Work in progress)*. IETF, Oct 2001.
7. M. Bhattacharyya, S. Hershkop, E. Eskin, and S. J. Stolfo. Met: An experimental system for malicious email tracking. In *Proceedings of the 2002 New Security Paradigms Workshop (NSPW-2002)*, Virginia Beach, VA, Sep 2002.
8. B. Bloom. Space/time tradeoffs in in hash coding with allowable errors. In *CACM*, pages 422–426, 1970.
9. A. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey. In *Annual Allerton Conference on Communication, Control, and Computing*, Urbana-Champaign, Illinois, USA, October 2002.
10. H. Bronniman, B. Chen, M. Dash, P. Haas, Y. Qiao, and P. Scheuerman. Efficient data-reduction methods for on-line association rule discovery. NSF Workshop on Next-Generation Data Mining, November 2002.
11. H. Burch and B. Cheswick. Tracing anonymous packets to their approximate source. In *Proc. USENIX LISA*, Dec 2000.
12. M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. In *ACM Symposium on Discrete Algorithms*, pages 635–644, 2001.
13. D. Dean, M. Franklin, and A. Stubblefield. An algebraic approach to IP traceback. In *Proceedings of NDSS*, Feb 2001.
14. H. Debar, M. Dacier, and A. Wepesi. A revised taxonomy for intrusion-detection systems. IBM Research Report, 1999.
15. P. Domingos and G. Hulten. Mining high-speed data streams. In *Proc. SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2000.
16. Sanstorm Enterprises. Netintercept. <http://www.sandstorm.com/products/netintercept/>, Feb 2003.
17. L. Fan, P. Cao, J. Almeida, and A. Broder. Summary cache: A scalable wide-area web cache sharing protocol. In *Proceeding of SIGCOMM '98*. SIGCOMM, 1998.
18. J. Frank. Artificial intelligence and intrusion detection: Current and future directions. In *Proceedings of the 17th National Computer Security Conference*, 1994.

19. P. Gibbons and Y. Matias. Synopsis data structures for massive data sets. In *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science: Special Issue on External Memory Algorithms and Visualization*, 1999.
20. K. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing wavelets on streams: one pass summaries for approximate aggregate queries. In *Proc. ACM Conf. Very Large Databases. VLDB*, 2001.
21. S. Guha, N. Koudas, and K. Shim. Data streams and histograms. In *Proc. ACM Symp. Theory Comput. STOC*, 2001.
22. G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proc. SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2001.
23. K. Ilgun, R. A. Kemmerer, and P. A. Porras. State transition analysis: A rule-based intrusion detection approach. *IEEE Transactions on Software Engineering*, March 1995.
24. H. S. Javitz and A. Valdes. The sri ides statistical anomaly detector. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, 1991.
25. S. Kumar and E. H. Spafford. An application of pattern matching in intrusion detection. Purdue University Technical Report CSD-TR-94-013, 1994.
26. A. Mankin, D. Massey, C. L. Wu, S. F. Wu, and L. Zhang. On design and evaluation of ‘intention-driven’ ICMP traceback. In *Proc. IEEE International Conference on Computer Communications and Networks*, Oct 2001.
27. G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *Proc. of the ACM Intl Conf. on Management of Data. SIGMOD*, Jun 1998.
28. G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets. In *Proc. of the ACM Intl Conf. on Management of Data. SIGMOD*, Jun 1999.
29. A. Mitchell and G. Vigna. Mnemosyne: Designing and implementing network short-term memory. In *International Conference on Engineering of Complex Computer Systems*. IEEE, Dec 2002.
30. R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma. Query processing, resource management, and approximation in a data stream management system. In *Proc. of the 2003 Conference on Innovative Data Systems Research (CIDR)*, Jan 2003.
31. V. Paxson. Bro: A system for detecting network intruders in real-time. *7th Annual USENIX Security Symposium*, January 1998.
32. P. A. Porras and Peter G. Neumann. Emerald: Event monitoring enabling responses to anomalous live disturbances. In *Proceedings of the National Information Systems Security Conference*, 1997.
33. T. H. Ptacek and T. N. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Secure Networks, Inc., January 1998.
34. P. Roberts. Nai goes forensic with infi nistream. In *InfoWorld*, http://www.infoworld.com/article/03/02/10/HNnai_1.html, Feb 2003.
35. S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. In *Proceedings of the 2000 ACM SIGCOMM Conference*, pages 295–306, Stockholm, Sweden, Aug 2000.
36. K. Shanmugasundaram, N. Memon, A. Savant, and H. Bronnimann. Efficient monitoring and storage of payloads for network forensics. Unpublished Manuscript, May 2003.
37. K. Shanmugasundaram, N. Memon, A. Savant, and H. Bronnimann. Fornet: A distributed forensics system. Unpublished Manuscript, May 2003.
38. 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 *ACM SIGCOMM*, San Diego, California, USA, August 2001.

39. D. Song and A. Perrig. Advanced and authenticated marking schemes for IP traceback. In *IEEE Infocomm*, 2001.
40. U. Thaper, S. Guha, P. Indyk, and N. Koudas. Dynamic multidimensional histograms. In *Proc. ACM Int. Symp. on Management of Data. SIGMOD*, 2002.
41. R. Winter and K. Auerbach. The big time: 1998 winter vldb survey. Database Programming Design, August 1998.
42. A. Yasinsac and Y. Manzano. Policies to enhance computer and network forensics. In *Workshop on Information Assurance and Security*, United States Military Academy, West Point, NY, Jun 2001. IEEE.