A Simple Framework for Distributed Forensics

Yongping Tang and Thomas E. Daniels

Department of Electrical and Computer Engineering

Iowa State University, Ames, IA 50011

{tangyp, daniels}@iastate.edu

***Abstract*—Networks have become omnipresent in today's world and part of the basic infrastructure. The safety problem is important and urgent for all the network users. But the current situation in this field is very severe – not only is it difficult to block network criminals but also in many cases unable to find them out. There is a growing need for systems that allow not only the detection of complex attacks, but after the fact understanding of what happened. This could be used in a forensic sense or simply as a managerial tool to recover and repair damaged systems.**

There are few network systems that support forensic evidence collection and the current systems also lack effective attack attribution. In this paper, we will provide a network forensics framework based on the distributed techniques thereby providing an integrated platform for automatic forensic evidence collection and efficient data storage, supporting easy integration of known attribution methods, effective cooperation and an attack attribution graph generation mechanism to illustrate hacking procedures.

***Index Terms*—Distributed Forensics System, Agent, Proxy, Attack Attribution Graph**

I. INTRODUCTION

NETWORKS have become ubiquitous and part of the critical infrastructure in today's world. More and more people rely on network for their work and study. Huge volumes of important information such as finance information, electronic money and digital signature are transferred through the Internet all the time. But at the same time, with the development of the network, threats and attacks also rose rapidly. These threats show that networks are vulnerable to attacks and misuse. Every year, companies lose a great amount of money due to network crimes. So safety of network has become a critical problem for everyone who uses the network.

To secure the network, generally, we have two aspects of network protection approaches. The first is using defensive mechanisms to prevent the network criminals. This type of approaches typically needs to find out the network vulnerabilities and then to make patches to block these vulnerabilities, or to apply some special facilities to block malicious communication from outside. One example of these facilities that is most commonly used is the firewall. But this defensive method cannot completely solve the network safety problems. First not only are we unable to find out all vulnerabilities, but also new vulnerabilities will always occur even when we patch old ones. In fact, the vulnerabilities in design and implementation of network are impossible to completely avoid. With the rapid development of network technology, the structure of the network becomes more and more complex, thus giving greater opportunity for introducing vulnerabilities. The increasing complexity of systems also makes vulnerabilities much harder to find. Second firewalls have limitations on the amount of state available as well as their knowledge of the hosts receiving the content and therefore cannot deal with the entire complex network environment.

As a result, other types of network protection approaches become more important. These approaches are not to block the network crimes but to find them, and to collect enough evidence of these crimes. Network criminals will be punished for their illegal actions thereby providing a deterrent to online crime. These methods are called network forensics. The main procedure of network forensics is to find the process of how the criminals execute their malicious actions and to collect evidence to support an investigation. An effective network forensic system may increase the cost of the network crimes for the attacker and thus reduce the network crime rates.

Many criminals are never been found despite causing great losses. Even when they are found, attackers may waste investigator's time and resources in manual examination of large log files. Without being punished for the lack of effective network forensics systems, criminals are un-apprehended. In fact, current network forensic investigations are often executed manually by examining different types of logs and using these logs to reconstruct the events that led to an attack. But due to the large volume of data in today's network, such manual methods make the analysis infeasible. We need infrastructures for forensic data collection, storage and analysis. Although there are many logging mechanisms and audit trails kept on hosts, these event logs only care about what happens on the local machines without considering the relationship between local hosts and other network nodes. We need a network system to share the forensic information and do the correlation analysis in an integrated approach. The current network forensic model

Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05)
1545-0678/05 $20.00 © 2005 IEEE

is also an after the process forensics procedure. Only after a criminal event happens, can such a model make a corresponding response through a long time and with human interventions. Further such a model usually cannot find the crucial evidence for network forensics because in the network environment digital evidence disappears quickly.

To solve all these problems, it is necessary to provide a distributed system that can monitor large networks. This distributed system can collect and store the forensic data simultaneously at different locations, and exchange these data in a properly managed way to support the forensic analysis. And because the heavy data flow on the network, it is infeasible to store all the raw data of the traffic. So a data preprocessor is needed, which would use a proper data structure to store the forensic data and disseminate these data in a proper quantity.

The manual network forensics model inevitably leads to long response time and missing the critical evidence. A practical network forensics system should be able to collect data in real time and use an automatic communication and coordination mechanism to provide a much quicker response to the adversary network events than possible manually.

There are very few existing network forensic systems, ForNet [4] is a good networks forensic system but it lacks of cooperative attack attribution. It is more like a network data collection and query system. In this paper, we will give a framework of a distributed forensics system. This system aims at providing a proper method to collect, store and analyses forensics information. It will also provide automatic evidence collection and quick response to network criminals. To find out the attacker's hacking path, this system will support attack attribution by integrating some known attribution methods and provide an attack attribution graph generation mechanism to illustrate the attack. In general, our work aims at providing three contributions to network forensics:

(1) An efficient data storage method based on distributed storage, pre-selection and specific data compression;

(2) A general framework using distributed techniques supports easy integration of known attribution methods and effective cooperation within the system;

(3) An attack attribution graph generation mechanism to illustrate hacking procedures.

In section III, we will give a detailed description of the system.

II. RELATED WORK

There are two related issues to this network forensic system: one is about the attack attribution techniques; the other is about different types of Intruder Detection Systems. This system aims at providing an open and scaleable environment to integrate some effective attribution methods and also utilize IDS as part of the system to filter network traffic.

One important type of attribution techniques has been studied is stepping stone correlation based on timing information. In [2, 3] several stepping stone correlation methods are discussed. [2] presents the correlation of ON/OFF periods of interactive packets. It is robust to retransmission but not effective to non-interactive traffic and packets with random delay, reorder and chaff. [3] gives active watermark tracing method by actively add watermark into the flows by manipulating inter-packet delays. It can deal with the packets random delay problem. But when the delay is too large and causes packets reorder, or when chaff added, it becomes ineffective.

Another studied attack attribution techniques are IP traceback and packet marking. In [1], a Source Path Isolation Engine (SPIE) was developed to do IP traceback and the Bloom Filter was used as the data storage mechanism. In [5], based on the topology of Internet, a new deterministic packet marking technique was provided and only total 52 bits would be used for marking the whole path.

At the same time, Intrusion Detection Systems (IDS) have been studied extensively over the past few years and the mainstream of work can be categorized into two groups [7], misuse detection and anomaly detection systems. Misuse detection is signature-based methods relying on a specific signature of an intrusion, which triggers an alarm after being found. Such systems cannot detect novel attacks because only known attacks have signature available. On the other hand, anomaly detection systems [6] rely on characterizing normal operations of a network or a host, and attempt to detect significant deviations from the norm to trigger an alarm. Although anomaly detection systems can detect novel attacks to a certain extent, they are difficult to be practical due to the so high false alarms. One solution [8] is to combine signature based systems and anomaly detection systems that can decrease false alarm rates. In this system, currently we use a lightweight IDS, called snort [9] as part of our integrated system as the data filter to find out the attack related data in the network traffic. Snort is a versatile, lightweight and very useful intrusion detection system. It can be used as a straight packet sniffer, a packet logger and a full-blown network intrusion detection system. The drawback here is that snort is still a misuse detection system and it cannot find novel attacks. We need enhance this function in the future.

III. SYSTEM DESCRIPTION

In this section we give a detailed description of the system architecture of our distributed forensics system. This system will be deployed over a large network (such as a campus network) and will store the data in a distributed way but do the data analysis in a cooperating method.

A. Agent and proxy based distributed system

The basic units of this network forensics system are agents and proxies. Agents will monitor the network that we need to do the computer forensics. Proxies are data processing center and the place where the attack attribution will be executed. We deploy agents among the network to collect data and the data will be saved locally on the agents. Proxies get pre-processed data from agents and do the forensic analysis.

Each agent will monitor a sub network (the size of this sub

Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05)
1545-0678/05 $20.00 © 2005 IEEE

network will be defined by network administrators) and we call such a sub network as an agent subnet. Several agents will connect to a proxy that controls these agents. All the proxies are interconnected with each other, and if possible, a proxy should be deployed on an exclusive host that will just communicate with agents and other proxies, not do any other communications. Through this approach, the data traffic between one agent and one proxy and the data traffic between one proxy and another proxy will not affect the normal traffic of the network.

Because one proxy will control several agents, it will then control several agent subnets. We call the collection of these agent subnets as a proxy subnet.

An agent only sends data to its control proxy. The connection between agent and proxy uses direct socket connection.

Proxies do not directly collect data from network. They just receive preprocessed data from agents. To avoid single point failure, we do not use a central control server but rather we treat proxies as peers. A proxy got processed data will do data analysis and cooperate with other proxies. The communication between proxies is based on a distributed environment instead of socket because the cooperation between proxies is more complex than just static connections. If all the proxies can work together as a whole system, many correlation methods can be used to a larger scope and get more precise results.

The following is the basic illustration of the system architecture. (Figure 1) Using this architecture, we can store the data in a distributed way and cooperatively process these data.

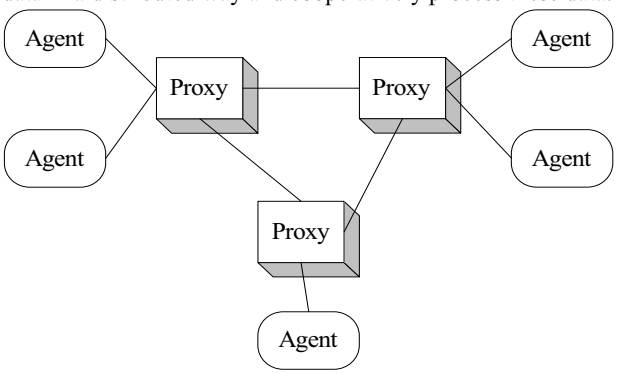

Figure 1. Architecture of the Framework

B. Agent

Agents are responsible for data collection and simple analysis. They will be deployed at specially chosen hosts and monitor an agent subnet. In our system, an agent has four functional parts: data collection module, data reduction module, data processing and analyzing module, and communication and control module. (Figure 2)

1) Data collection module

The data collection module is used to collect data from the network. Data collection will use both active and passive methods. The active method is to use a probe to periodically scan vulnerabilities and to get the data; the passive method is to listen on the network interface and capture the raw traffic data based on some criteria. This is the first step to get the forensic evidence and is also the most important step to ensure that our system can get the evidence in real time. Based on the active and passive methods, the data collection module has two parts:

Network scan probe: this probe will scan the agent subnet for vulnerabilities based on known patterns of the discovered vulnerabilities. If the probe finds some vulnerability, the agent will report this information to its control proxy. The proxy will automatically inform the network administrator. Also when attack is found, proxy will pay more attention to the activities on the hosts with vulnerability. The probe will scan the subnet at a fixed time interval and the pre-configured pattern can be changed when new vulnerabilities disclosed.

Traffic data capturer: The traffic data capturer uses libpcap [11] to capture the network raw packet. It has a built in packet parser. The parser will identify each part of a packet and then the capturer will decide which parts need to be saved based on the dynamic pre-selection criteria. We know for most forensics problems only the packet header is needed for analysis. For example, the data for stepping stone analysis only need the timestamp of packets, basic connection information such as 4-tuple (source IP address, source TCP port, destination IP address and destination TCP port). Sometime, the sequence number, the acknowledge number and the flags in TCP header are also needed. So based on the pre-selection criteria, it only captures the needed part of a packet.

2) Data reduction module

One major difficulty for all network-logging systems is that the huge volume of the traffic data and privacy issues. It is impossible to store all the raw traffic data directly. One method used widely is Bloom Filter. It uses hash functions on routers to record the transfer path of a packet and using little storage space. The method works well for IP trace back systems. For the forensics system, Bloom Filter has disadvantage in evidence collection. Based on the hash functions, Bloom Filter can answer questions like something happened or not (in fact, Bloom filters are probabilistic, the only certain thing they can tell us is whether an event has not happened, any positive responses are all based on probabilities), but it cannot answer questions like what has happened, because it only records the hash of events, the hash in nature cannot convert back to events. But for forensics system, in most situations it is necessary to answer questions like what has happened. To know what has happened as forensic evidence, the system still need store the history network events directly.

In our system, we use three methods to deal with the data storage problem. The first method is the distributed data storage provided by the system architecture. Agents just store their local data and the data only relate to a small subnet. Thus the data volume at one site will not be very large. The second method is the pre-selection criteria we mentioned above. It tells the agent only store the needed data. The third method we use is data compression using arithmetic coding[10].

We know most of the network use Ethernet and TCP/IP protocols and most of the network traffic packets are Ethernet

Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05)
1545-0678/05 $20.00 © 2005 IEEE

packets. Network traffic data captured by libpcap library has a fixed format called TCPDUMP format, which consists of length indicators, packet timestamp and the Ethernet packet, which means that the data we captured has some specific structures. And because this data reduction module stores the data in compressed format to hard disk directly and continuously (may be as long as the work time of the system), and there is no mid-step for compressing the existed data file, all the raw data can be read only once. So the data here have two characteristics: specified data structure and one-time scan. We choose arithmetic coding to do the compression based on these two characteristics.

Arithmetic coding is a data compression technique originated from Shannon techniques. And it can get a best compression rate in theory. The basic idea of arithmetic coding is to assign a range to each symbol based on its probability within a probability line [0, 1]. It completely bypasses the idea of replacing an input symbol with a specific code. Instead, it takes a stream of input symbols and replaces it with a single floating-point output number. The longer (and more complex) the message, the more bits are needed in the output number. Due to the high cost of floating-point operation, all implementations of arithmetic coding use integer arithmetic to achieve a faster speed.

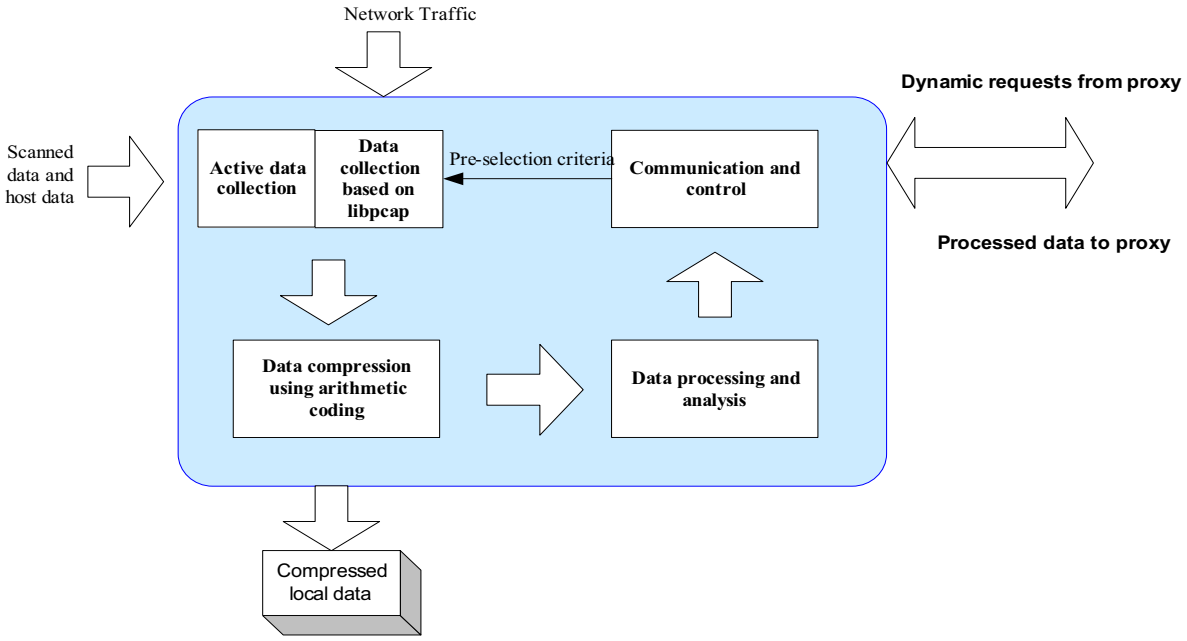

Figure 2. Structure of Agent

Arithmetic coding needs statistic model for data when doing the compression. We develop different models for different parts of the data and then use the basic coding engine to do the compression. The timestamp has 8 bytes long. The first 4 bytes are seconds since Greenwich led the Western world out of the 1960's. The low 4 bytes are the absolute number of microseconds. Because 3 bytes are enough for the maximum value of the microseconds (999999 = 0x0f423f), the highest byte in these 4 bytes is 0. And it is clear that the arrival time of the packets is chronological, so the timestamp is in an increasing order. Thus the entropy of the timestamp is much less than 64-bit.

We develop a model called *TimestampModel* for the timestamp. This model will encode the whole first timestamp (7 bytes, the byte equals to 0 is omitted). From the second timestamp, this model computes the delta value from the previous timestamp. If the delta only use 4 bytes (1 byte for the difference of seconds and 3 bytes for the difference of microseconds), this model will adaptively encode these 4 bytes; otherwise, it falls back to encoding the 7 bytes long whole timestamp and begin a new loop repeated the above procedure.

There are two length indicators in the TCPDUMP packet, 4 bytes long each. But only 2 bytes in the 4 bytes are used because the Ethernet packets are not longer than 1514 bytes (not include the 4 bytes CRC checksum). The model for the length indicator that we develop is called *LengthIndModel*. It first omits the zero bytes. Based on our experiments, we found most packets are shorter than 128 bytes. The *LengthIndModel* model uses a cumulative history table with the alphabet 0 to 127. If the packet is longer than 127, the model takes 2 symbols for one indicator and a prefix symbol zero. For example, if the packet length is 56, we use the symbol 0x38. If the length is 1000, then 3 symbols will be used: 0x00, 0x03, and 0xE8. Although three one byte long symbols are used for two bytes long length, the probability of two bytes long lengths is much smaller than that of one byte long length, such an encode method is still efficient.

There are two MAC addresses in the Ethernet packet header. The first is the destination machine address and the second is the source machine address. We know that the MAC addresses in the Ethernet packet header will be changed at each machine. So if we capture packets at a certain machine, the MAC

Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05)
1545-0678/05 $20.00 © 2005 **IEEE**

addresses we will meet are only the address of this machine and the addresses of the machines that directly connect to this machine. So the total number of the MAC addresses is limited and at most time, it is not very large. The *MacModel* use a dynamic allocated structure to save the MAC addresses. When it meets a new MAC address, a new node is allocated and the frequency of this MAC address is initialized to 1. If a MAC address already existed in the dynamic table, only the frequency will be increased 1. When do the encoding, a one-byte long symbol will replace the MAC address to be encoded. If the total number of MAC addresses in the dynamic table has exceeded 256 (in a large network), we need to refresh our model to ensure that we can still use one byte long symbol present the 6 bytes long MAC address. Through the above procedure, we can greatly reduce the entropy of the MAC addresses by first use much shorter symbols and second use arithmetic coding compression.

Other models for TCPDUMP data we developed are Ethernet type model, IP address model, ARP header model, IP header model, UDP header model, TCP header model, ICMP header model and payload model. These models, using arithmetic coding, can achieve a much better compression ratio than the most commonly used compression tools such as gzip and the space needed for storage is much less than that the raw data need. The following table shows some of the results using arithmetic coding to compress the tcpdump format data.

Timestamp	Gzip-9 size	AC size	AC Ratio
170520	54779	36766	21.56%
7526400	2237381	1409781	18.73%
15776784	4824212	2924257	18.53%
LengthIndicator	Gzip-9 size	AC size	AC Ratio
170520	10659	7168	4.20%
7526400	495433	309760	4.11%
15776784	1218123	740352	6.69%
MAC Address	Gzip-9 size	AC size	AC Ratio
255780	3330	2623	1.02%
11289600	178627	125128	1.11%
23665176	408525	287773	1.21%
Ether Type	Gzip-9 size	AC size	AC Ratio
42630	384	164	0.38%
1881600	2549	1216	0.06%
3944196	4614	2228	0.05%
IP header size	Gzip-9 size	AC size	AC ratio
420540	131954	97475	23.18%
18804880	5543238	3496764	18.59%
39431040	12229471	7819308	19.83%
ARP header size	Gzip-9 size	AC size	AC ratio
10656	167	69	0.64%
20572	242	172	0.83%
TCP header size	Gzip-9 size	AC size	AC ratio
560240	203796	126926	22.65%
20134856	9010259	7551694	37.51%

Table 1. Compression Results by Arithmetic Coding

3) Data processing and analyzing module

The major responsibilities of the data process and analysis module are data pre-processing and data preparation. Although we use data reduction techniques to reduce the volume of the saved data, it is still impossible for an agent to send all its data to its control proxy. Our goal is to reduce agent-proxy connections as much as possible. This module will use a lightweight IDS such as snort to filter and pre-process the traffic data. If there is an attack in the captured data, snort will generate many alerts. Most of the alerts are repeated and can be combined together. Here we use an algorithm called *Leader-Follower* Algorithm to combine the raw alerts into hyper alerts. This algorithm is very simple, it reads the raw alerts one by one and groups them to some hyper alerts. The criteria for grouping are alerts have some source address, destination address and attack class; alerts' start time and end time within a same time-window. After finishing data pre-processing, the agent will send the hyper alerts to the proxy to indicate there is an attack found. We call this procedure as *the attack-driven message transfer*. Also, sometimes the proxy will need more detail data from the agent. When the agent gets such requests from its control proxy, it will try to answer the query. For example, when a proxy wants to do a stepping stone analysis, it needs the necessary information from the agent. The agent then decompresses the saved data and extracts the data based on the TCP 4-tuple and time range. The output data will include TCP connection 4-tuple (source IP address, source TCP port, destination IP address and destination TCP port) and the packet timestamps within the connection. These data then send back to the proxy. We call such procedure as *the query-driven message transfer*.

4) Communication and control module

Communication and control module is the hinge between an agent and its control proxy. This module sends data to and receives data from the agent's control proxy through socket. When it needs to send data, it will connect to a specific port on the proxy and then send the data. It also has a server socket. When the proxy sends some requests to the agent, the proxy will connect to this server socket.

When the agent receives data selection requests (the proxy will generate these requests based on its history data needs), this module will convert the requests into pre-selection criteria and send the criteria to the data collection module. When the agent receives data query requests, this module will send the query requests to the data processing and analyzing module to do data preparation.

C. Proxy

Proxies are connected with each other and constitute an integrated platform for network forensics. As we mentioned before, the proxies need to work cooperatively and the communications among them may be very complex, so we use CORBA to implement the communication mechanism instead of a direct socket connection. All the proxies can work together

Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05)
1545-0678/05 $20.00 © 2005 IEEE

seamlessly like a single node. The data analysis methods will be developed as some services deployed on the proxies. When one proxy encounters a forensics issue (e.g. it gets some processed data from its agents and the data indicates an attack) and needs to do some analysis, it will call the remote services on related proxies. These remote services will analyze their own local data and the results will be disseminated via remote calls. In this way, data correlation can be done in a network environment instead of being limited to one machine at the same time.

Based on the distributed techniques, this system is an open and a scaleable environment. For the openness, we can easily add new data analysis methods to this system just by deploy them as new CORBA services. For the scalability, we can apply this system to a large network with more proxies or apply this system to a small network with fewer proxies. The goal of our system is to provide an environment that can use different data analysis methods to do the attack attribution according to different situation and collect the attribution results as the forensic evidences.

1) Attack attribution through proxies

We define attack attribution graph G = {E, V} for our system. Here V is the set of proxies that are related to the attack attribution, i.e. if one proxy is in V, it means something happened in this proxy subnet is related to the attack. E is the set of some connections that belong to the attack. For example, if there is a telnet from proxy subnet A to proxy subnet B to do the illegal access, the attack attribution graph G then have two vertices A and B and a directed edge (A, B) with an attribution illegal telnet.

The procedure to generate the attack attribution graph proceeds roughly as follows: when a proxy (the victim is in this proxy subnet) finds an attack from a remote proxy subnet, it will inform the remote proxy and share the related information. The remote proxy then checks the related information on its own historical data. Some special connections such as telnet, SSH and ftp will be concerned based on a time window because attackers often use these services.

If suspicious events found, the remote proxy will contact other proxies if they are related to these events. And then such activities continue to other proxies. All the proxies will work simultaneously to generate a sub-graph of its own and the final attack attribution graph is then generated by combine all the sub-graphs. The graph combination algorithm can be described as the following: During a cooperative analysis, if proxy X contact proxy Y for help, we say X is Y's anterior proxy and Y is X's posterior proxy. Initially, the proxies without posterior proxy send sub-graphs to their anterior proxy. When a proxy gets graphs from its posterior proxies, it combines these graphs and deletes repeated edges and vertices. Then it sends this new graph its anterior proxy. The combination procedure terminates at the original proxy that found the attack. If one proxy has more than one anterior proxy, it need decide which anterior proxies is important based on its current graph then send graph to the important anterior proxy. The final graph may not fit the real attack completely, but it should give a lot of useful information about the attack that took place.

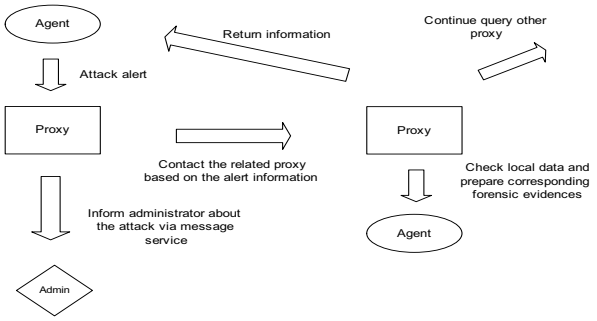

Figure 3. Workflow of the Proxies

One useful data analysis method to be deployed on the proxies is stepping stone analysis. Stepping stone analysis is very important for attack attribution because almost all attackers will not directly attack the victim but use some intermediate hosts as "stepping stones" to hide the origin location. One such method is the timing-based algorithm that is described in [2] to do the stepping stone analysis. This algorithm will compute the ON/OFF pattern based on the connection information got at one machine. By the distributed CORBA services, this algorithm will run on different proxies for different but related data simultaneously. The resulting ON/OFF pattern will be shared and stepping-stones can be discovered during the cooperation of the proxies. The coordination among these proxies is seamless.

2) Sample Scenario

Here we use a sample scenario to illustrate how the proxies work cooperatively to generate an attack attribution graph. The scenario is executed in our lab's test bed and described as the following:

The test bed will have 2 subnets with 3 hosts each. Each subnet will have one proxy and one agent. The agent will not capture the traffic data if the data are to or from proxy. We call one subnet as subnet A because it has proxy A and the other subnet is B because it has proxy B. We make one host in subnet B as the victim and it has a Windows DCOM buffer overflow vulnerability. We make one host in subnet A as the stepping-stone. The attacker will first login into the stepping-stone through telnet. And then he will download a hacking tool from a public FTP site and use this hacking tool to gain illegal access to the victim.

The agent in subnet B will find the attack because it will monitor the traffic in this subnet. Then the agent will send the hyper alerts to proxy B based on the attack-driven message transferring. Proxy B will tell the attack information to proxy A in another subnet because the attack is from the subnet managed by proxy A. When proxy A receives these attack related data, it will then check its data (by query its control agent) about some specific service. Here it will check the telnet connection, ftp connection and SSH connection information in its history data using on a time window. Then the proxy A will find there are a telnet and an ftp connection suspicious.

At the same time, proxy B will also inform the proxy A to do ON/OFF analysis by sending it the stepping stone information.

Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05)
1545-0678/05 $20.00 © 2005 IEEE

Proxy A will query the data from its agent based on the time information provided by proxy B. Proxy A will check all the input connection that have a similar time scope provided by proxy B and then get the all the related TCP connection information from its agent. Then it does ON/OFF analysis for all the connections it chooses compare to the series from proxy B. Then the proxy A will find the telnet from outside is a stepping stone connection.

Proxy A then uses the suspicious ftp information and the stepping stone results as well as the attack connection to construct a graph. And proxy B will also construct graph to indicate the attack from proxy A. Proxy A will send its graph to proxy B and proxy B will combine all the graphs together in the end. The final graph will give the attack attribution path for the attack it finds. The process to construct the graph is in fact a distributed algorithm running on different proxies. Each proxy will use its local attack attribution result to construct a sub-graph and finally all sub-graphs will be combined together. The process for this sample scenario is like the following:

For proxy B, it only knows an attack from proxy A (in fact, from one host managed by proxy B to one host managed by proxy A). So it just constructs a graph with two vertexes: v presents proxy B and u presents proxy A. And a directed edge from u to v presents attack from u to v.

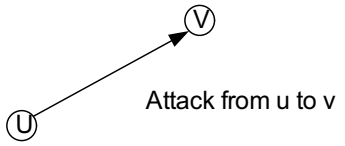

Figure 4. Sub-graph by Proxy B

For proxy A, the situation is a little complex because it has two suspicious connections: a stepping stone connection, ftp connection. So it will construct a directed graph with four vertexes: u presents itself, v presents proxy B, x presents the stepping stone source and y presents the ftp server. Directed edge (u,v) presents it has an attack to proxy B; (x, u) presents the ingoing stepping stone connection and (u,y) presents a ftp get from a ftp server somewhere.

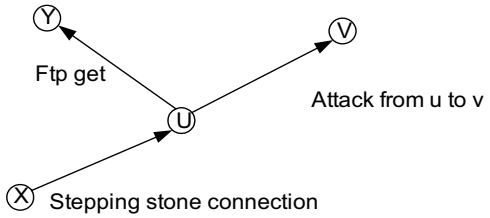

Figure 5. Sub-graph by Proxy A

Then the two graphs are combined at proxy B to generate the final graph. In the end, a report about this attack will be generated by proxy B and it then save the related data and send an email to the administrator with this report by CORBA inform service.

IV. FUTURE WORK

There is much remains to be done to implement this framework. First, We have not considered the security of the agents yet. We need further work to deal with the compromised agents. Second, the current system is just a simple implementation and many other attack attribution methods can be added to the proxy. Also, when new methods are added in, the communication between proxies will need to be redesigned to avoid unnecessary remote call. The third is that the data analysis and process module in the agent need further consideration. The data filter is now based on a lightweight IDS. But the IDS suffers from some known error conditions, so we must enhance the data analysis and process module in the agent. The fourth thing is that we need more consideration on the process to generate the attack attribution graph. Because one host will have many connections at the same time, how to judge a connection that is attack related is complex. To find the real attack connections, we may need more analysis based on the history data and network model. One possibility is abnormal detection. If some connections do not follow some patterns seen in history data, these connections are highly suspicious.

REFERENCES

[1] Alex C. Snoeren, “Single-Packet IP Traceback” in IEEE/ACM Transactions on Networking (ToN), 2 Volume 10, Number 6, December 2002. Pages 721-734.

[2] Yin Zhang, Detecting Stepping Stones. Http ://www.icir.org /vern/papers/stepping/.

[3] X. Wang and D. S. Reeves, Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays, in Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS 2003), Washington DC, USA, Oct. 2003

[4] Kulesh Shanmugasundaram, Nasir Memon,etc, ForNet: A Distributed Forensics Network. http: // isis . poly.edu/projects/fornet/docs/talks/mmm-acns-2003-talk.pdf.

[5] Basheer Al-Duwairi and Thomas E. Daniels, Topology Based Packet Marking.

[6] Fabio Gonzalez, Jonatan Gomez, Madhavi Kaniganti and Dipankar Dasgupta, An Evolutionary Approach to Generate Fuzzy Anomaly Signatures, http://issrl.cs.memphis.edu/papers/ais/2003/IEEEIAW.pdf

[7] Earl Carter, Intrusion Detection Systems, http://www.ciscopress.com/articles/article.asp?p=25334

[8] Kaleton Internet, Combination of Misuse and Anomaly Network Intrusion Detection Systems, http://packetstormsecurity.nl/papers/IDS/kaletonidspaper.pdf

[9] Brian Caswell and Jeremy Hewlett, Snort Users Manual, http://www.snort.org/docs/snort_manual/

[10] Arturo San Emeterio Campos, Arithmetic coding, http://www.arturocampos.com/ac_arithmetic.html

[11] Tim Carstens, Programming with pcap, http://www.tcpdump.org/pcap.htm

Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW’05)
1545-0678/05 $20.00 © 2005 **IEEE**