Packet Tracker
Final Report*

CERIAS Technical Report 2000-23

Florian Buchholz
Thomas E. Daniels
Benjamin Kuperman
Clay Shields
CERIAS
Purdue University
1315 Recitation Building
West Lafayette, IN, 47907-1315
{florian,daniels,kuperman,clay}@cerias.purdue.edu

*Portions of this work were su pported by sponsors of CERIAS.

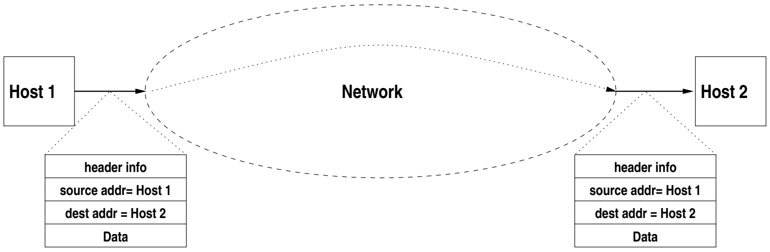

Figure 1: Simple Network Model

1 Introduction

When creating the suite of protocols that are used in the Internet today, the designers were more concerned with ensuring reliability and survivability than they were with providing accounting or security services [11]. This lead to a very simple network model. First, a packet-switched network was chosen to allow robustness and ease of routing around failures in the network. Second, all the network would provide was a simple packet-delivery service. This model was the basis for the Internet Protocol (IP), the fundamental protocol used in the Internet [27]. While in IP there are a few options for specifying a particular type of service requested from the network, and options to record the route the packet traveled or to mandate a particular route for the packet, all other services — including reliable transmission, congestion control and authentication of the source of a transmitted packet — have to take place at the endpoints of the communication [27]. Under this simple model, a host connected to the network gives a packet to the network, and the network attempts to deliver it to the given destination address. This is shown in Figure 1, in which Host 1 sends a packet to Host 2. As shown, the contents of the packet are some header information (including the packet and header lengths and checksum, the protocol being used and the type of service desired), the source of the packet and the destination for which it is intended, and data (which includes information not only for the application but also as necessary for multiplexing and reliability).

1.1 The Address Spoofing Problem

While this simple model formed the basis for the wide variety of successful services extant today, it is not without its flaws. Based on the information readily available, a host cannot be sure that a received packet has the correct source address. While in some cases the source correctness may be inferred from other data in the packet, particularly if some sort of strong authentication is used, it is typically very easy for some malicious sender to spoof the address of a packet that it sends. Figure 2 shows a malicious host sending a packet to Host 2 while pretending to be Host 1.

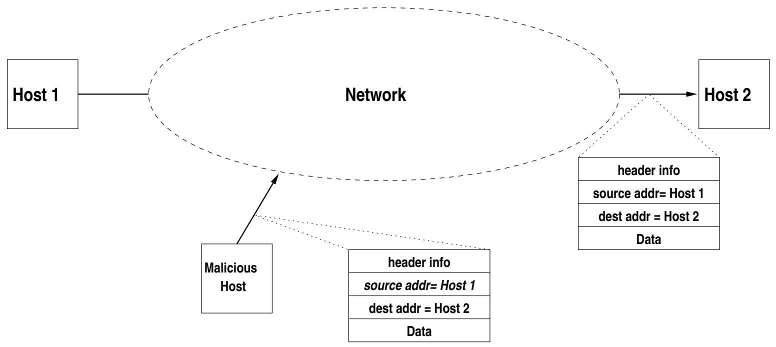

Figure 2: Address Spoofing

This capability has been used in a number of attacks, either to gain access to a host by exploiting a trust relationship it has with another host based solely on IP addresses [19, 2, 13], or to perpetuate a denial-of-service attack [6, 5, 7, 3]. In response, some individual domains have voluntarily added filters to their outgoing router that drop outgoing packets with external addresses. This prevents users inside the domain from spoofing packets by limiting the range of addresses that can be forged to those within that domain but does not prevent the use of address spoofing within the domain to hide an insider attack or to exploit internal trust relations.

It is also possible to prevent some of these attacks at the receiving end by requiring use of strong authentication, but that is not yet consistently feasible in practice, as it may be difficult to require such authentication for small packets as TCP SYNs. It can also be computationally expensive in terms of key management and key exchange. More importantly, while authentication will cause rejection of spoofed packets, it does not allow for discovery of the attacker, who is difficult to track and locate as the packet source address does not reflect any information as to his location. Finally, an attacker who has compromised a host may have access to the key information needed to defeat the authentication mechanism thereby leaving nothing more with which to trace the attack than in the unauthenticated case.

Currently, the main method used to locate such an attacker is to attempt to trace back the stream of forged packets while the attack is active [18, 31, 8, 23]. By following the stream of packets from router to router within the network it is possible to trace back and locate the particular source that might be conducting an attack. This is shown in Figure 3, where the internals of the network are revealed to be a number of *routers*, which are specialized hardware devices that do packet routing and forwarding, and the traceback occurs through the routers in the order they are numbered. This method is very limited, however, as it is necessary to have access to all routers along the path from victim to attacker, and this is often not the case. The attacker's packets may be traversing a number of domains under

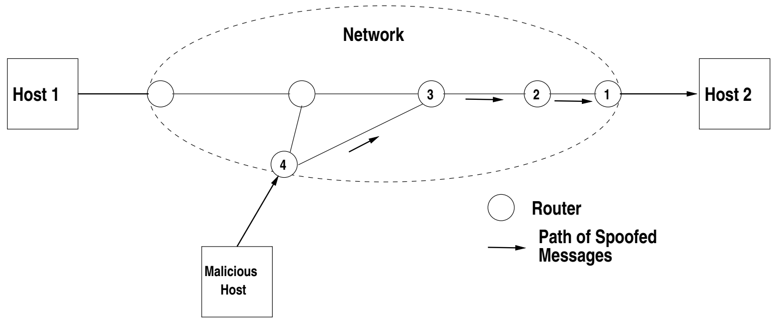

Figure 3: The Route of Traceback

different administrative control, in which case it is necessary to contact other network administrators, who have other demands on their time and may not be able to respond to an attack against a target for which they are not responsible. Additionally, this method is limited to tracing active attacks and thus must be done while the attack is occurring, or in the case of Intrusion Detection and Isolation Protocol (IDIP) [23], shortly thereafter. In all of these systems except for IDIP, no state is maintained in the network and therefore it is impossible to trace an attack after it has completed. In IDIP, a small amount of state is kept at special routers installed throughout the network that allows tracing of a packet immediately after its reception. It is unclear what level of state may be maintained without overburdening network components and how long this window of traceability is.

1.2 Traceback of Streams

An attacker may also take other actions to hide his location. A common (and unfortunately, often easy) way to do this is to compromise some remote host and use it to launch attacks. This makes it difficult to locate a particular attacker, because even if tracing back a stream of spoofed messages is successful, it results only in the location of the compromised host. The trace back then might have to be repeated if the audit data in the compromised host has been corrupted, or is insufficient to determine where the attacker came from. An attacker might use a series of compromised hosts, making the process of locating him very difficult, because hosts in multiple domains may be involved in different political regions around the world, or because the attacker may not be actively connected to the compromised host that is launching the attack, having set up the attack program to run after he has disconnected. Figure 4 illustrates how an attacker can use this method to hide their location. Notice that the data stream from the attacker passes in and out of the network at several different places.

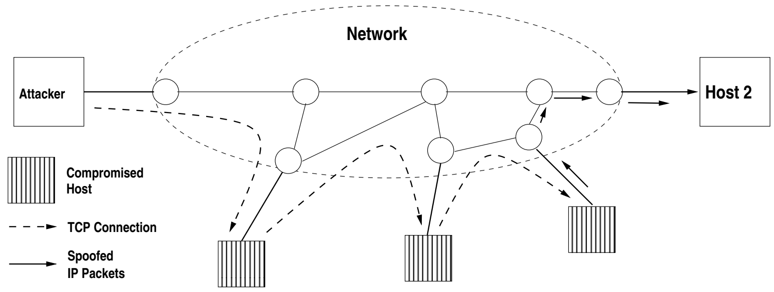

Figure 4: Using Compromised Hosts to Conceal Origin

1.3 The Problem

In each of these cases an attacker is aided by the fact that the network and its hosts do not deliver or maintain any information about the traffic carried. Packets are not delivered with any information about their originator, and this results in an attacker being able assume the identity of others, in terms of sending packets, with relative impunity. Data streams are traceable only while they are active, and once ended are impossible to follow, making it possible for an attacker to escape without his location being detected.

1.4 Past Work

There is previous and current work that attempts to characterize how a network device that sits at the edge of a particular network (either at the sub-net the host lies on, or at the boundary between autonomous domains) can attempt to match an incoming and outgoing stream so as to detect when an attacker is using a compromised host for forwarding. This serves two purposes. First, it allows for detection of compromised hosts. Second, it allows for a "shortcut" in any attempt to trace back a stream to an attacker. While most of these previous efforts have been attempts to detect an attackers activity in real time [18, 31, 8], some work has been done on recording and providing a fingerprint of a data stream, so that streams monitored in different locations around the network can be compared after the fact [24, 25].

Other work presents a host-based approach to tracing an attacker who is logged on through a number of hosts[14]. During the process of logging into a remote host, the originating host presents a trace for the user showing the hosts he has traversed and user names used on those hosts. The destination host then takes steps to verify that the user is actually logged into those hosts. If the verification step succeeds, the login is allowed. In either case, the trace is logged for later use by an administrator. In tightly controlled environments this may prove to be a useful approach, but it may be subverted using covert channels and other tricks.

Previously researched solutions are mostly unproven in real networks and have many problems that limit their utility. For instance, it is unclear how commonly false matches will occur in the fingerprinting techniques.[25, 24, 18]. Additionally, these techniques are susceptible to link-based encryption and evasion techniques similar to those described by others. [20] None of the techniques have addressed the problem of interdomain tracing nor incorporated measures to help assure the privacy of users. Also, none of the prior work has looked at limiting the ability to trace connections to authorized individuals. Finally, most of these traceback techniques only work for active attacks, but often attacks are not detected until it is too late to launch a trace during the attack.

Tracing packets and streams in a variety of network environments is an important component of the fledgling field of network forensics. The techniques for traceback systems proposed so far are only applicable to closed, tightly controlled environments. For traceback systems in open networks like the Internet, we must address the problems of privacy, trace integrity, and passive tracing. Furthermore, we must evaluate existing fingerprinting techniques for use in large, highly-connected networks and develop better techniques if necessary.

1.5 Anonymous Protocols

There have been a number of methods that have been used to maintain anonymity in the Internet, some for malicious purposes and some with the purpose of maintaining privacy. All the currently proposed methods make use of either address spoofing or use other hosts as proxies to hide the originator of a message, and thus simply take advantage of existing problems to hide the identity of the originator of a message. The use of address spoofing is more strongly associated with malicious attacks [19, 2, 13, 6, 5, 7, 3], that do not require two-way communication, as replies to the message are not routed back to the attacker but instead to whatever spoofed address is included.

Initiators using protocols designed for anonymous two-way communication are not able to use address spoofing effectively, and therefore relay their streams through other hosts in order to hide their identity. In the simplest of these schemes, a single proxy is used to "bounce" the stream to the destination. A malicious way to accomplish this is to use the forwarding mechanisms offered by some ftp servers [4]. A similar service is actually provided commercially by companies who serve as a relay for customers who want to anonymize their web browsing [30, 1]. These services use a single proxy to hide the IP address of the web browser from the web server, but that IP address is know to the proxy.

Because in some cases a single proxy may not be trustworthy, some other protocols make use of multiple proxies so that the original sender is anonymous with respect to the group of proxies. In these schemes, a trade-off between bandwidth consumption and anonymity is achieved by forwarding the streams of other in exchange for others forwarding your traffic [21, 22]. By using multiple proxies, no one proxy will know the originator of a stream as even though they can see the host that sent them the stream, the proxy will not know if the sender was the originator or some other forwarding proxy. In this case, it is possible to determine the originator of a particular stream if all traffic to or from a suspected originator is monitored. Outgoing streams that do not have a corresponding input stream must be originating at the monitored location. To make the matching of incoming and outgoing streams more difficult, these protocols use encryption and re-encrypt the stream between each link. This makes matching of the stream contents difficult, particularly if a particular host or proxy is forwarding multiple streams. Against these encrypted streams it is possible to use timing attacks that look at arrival and idle times to matching streams but no formal work describes this area. To preserve anonymity in the face of a timing attack, someone who wishes to preserve anonymity can use a more powerful proxy called a *mix* [10]. A mix takes in a number of connections and re-orders them before sending them back out. This adds latency to the connection, but makes it difficult to apply timing attacks to any particular mix. Fortunately from the point of view of being able to trace connections back, mixes are not commonly used for real time communications but are instead more commonly used for things such as electronic mail. Application to streams of data is certainly possible, but it is not being done at this time.

Existing protocols for anonymity take advantage of the ability to spoof IP addresses and the fact that it is difficult to trace a stream through multiple hosts. As techniques are developed to locate the originator of a single packet and to trace streams through the network, these protocols will provide a lesser degree of anonymity. It is easy to envision that as attackers and those concerned about network privacy lose their anonymity, however, that other forms of anonymous communication may be either discovered or developed, then deployed.

2 Environments

When examining possible systems of network traceback, it is important for us to define the environment in which our solutions will be applied. We suggest that there are two defining characteristics for solution environments: who controls the hosts and who controls the network.

Centralized host control implies that a single administrative authority is able to define and control all of the participating hosts on a network. This authority can determine the hardware, operating system, software installed, network services offered, and has the ability to customize or

	Network Control	
	Local	**Diverse**
Centralized Host control	Closed Model	Intranet Model
Diverse Host control	Academic	Internet

Table 1: A matrix of various computing environments encountered in a networked system of computers.

modify the network applications in any way they desire, provided the network will still carry their data.

Diverse host control implies that there is no central authority that can control and regulate the hosts connected to the network. No guarantees can be made about the specific hardware, software, operating system, or network services that hosts offer. Subsets of hosts might be under a single control, but not necessarily the entire set.

Local network control implies that all of the network infrastructure is under a single administrative domain. This administration can dictate the hardware, topology, and routing used in the network. This administration also has the authority to change network protocols, modify or examine any data flowing over the network, and regulate who or what is connected to the network.

Diverse network control implies that no central authority exists that controls the network infrastructure. Standard network protocols are required, or else data may or may not be routed. It is not possible to make any guarantees about who sees any data, or who the sender of any data is.

As seen in table 1, these four characteristics serve as the primary distinguisher for our four network models. The labels we have selected are arbitrary, and intended to convey the general concept of each environment.

Closed Model - This is an environment where the network hardware and all machines connected to it are under a single administrative control. This environment allows arbitrary changes to be made to network protocols and end machines. All of the packets viewed on the network should have been generated by a machine under the administrative control, and the packets never cross "untrusted" hardware.

Intranet Model - This environment is a collection of LANs that are interconnected by some form of "secure" connections (e.g. VPN tunnels, leased lines, etc.). Packets travel between clusters of machines across a shared network. A single entity can be controlling these clusters of machines, but it is necessary that the data be able to be carried over a public network if necessary, so the freedom to modify network topology, hardware, or protocols is lacking.

Academic Model - This environment is the situation on many University campuses. A single network connects the various machines on the campus, however, the machines are not centrally administrated. Any changes in the network protocols requires consensus building amongst the diverse groups. It is assumed that machines can be connected to the network at any time, and that they may or may not be well behaved.

Internet Model - There is a collection of LANs, WANs, and single hosts all sharing a network structure that does not have any central controlling authority.

The primary factors described above are not the only factors that define the environment in which traceback solutions are implemented. The following factors describe some of the various issues that also need to be considered, but are not fundamental to the enviroments in our discussions.

Resources of an entity are composed of three distinct subcomponents:

1. **Financial resources** describe the ability of an organization to purchase or otherwise expend money. This resource is used to augment and offset any deficiency that might exist in the other types of resources. Depending on the organization, the use of financial resources might be tightly controlled and/or under specific restrictions. For instance, the DoD can spend its allocated resources with relative freedom, but a small business might have to justify every single dollar of expenditure and use of this resource would be difficult.
2. **Technical resources** are sources of technical knowledge and expertise. If an organization does not naturally have a large technical reserve to draw on, they can expend financial resources to improve it by either hiring new staff or consultants. A University might have restrictions on the expenditure of financial resources, but they have a vast technical resource in their professors and graduate students.
3. **Manpower resources** expresses the ability for work to be done. A small company has limited manpower resources, while a university has a large set of manpower (students!). Again, any lack in this area can be offset by the expenditure of financial resources.
4. **Infrastructure resources** are the various computer and networking hardware available to an organization. This can be ex-

pressed in terms of bandwith, CPU power available, and hardware availability. An infrastructure rich environment such as the labs at Sun Microsystems or Cisco can build custom hardware to meet task needs, while an infrastructure poor organization, like a public school, would be hard pressed to meet current user needs.

Expectations of Privacy are defined in respect to some outside party. Users in an network environment might have no expectation of privacy as in a corporate setting where the users sign away their privacy. Another example is a classified computing environment where a user expects privacy from his peers so as to maintain confidentiality but certainly not from management for reasons of oversight. Users may also expect privacy from host and/or network administrators in some environments.

Societal Cost is a term that we use to describe the various incentives that exist for needing a network connection to be traced. In an intelligence agency, being able to trace an attacker's connection might be an issue of life and death of operatives. In a university setting, the issue might be to track an attacker that is using university resources to launch attacks. A provider of high bandwidth communication channels might not have any desire to trace connections, but simply need to bill the proper customers based on traffic.

The model environments described above serve as a baseline for evaluating and designing traceback systems. The defining characteristics of the environments limit the possible solutions for a given environment, and the secondary factors help us to further describe the approaches based on their social, financial, and practical impacts.

3 Conclusions

There are two main problems that make tracing network traffic to its source difficult: address spoofing and the redirection of traffic through multiple possibly compromised hosts. Each of the existing network forensics or traceback techniques addresses a small part of the overall problem space, but they fail to address many issues needed in a useful traceback system such as privacy and cross-domain traceability. Anonymity protocols take advantage of the very same problems that make it difficult to trace network traffic, and if used in the same environment as a traceback system, they imply a tradeoff must be made between traceability and anonymity. In order to evaluate possible solutions we will consider solutions in terms of appropriateness for a model environment and also using various secondary factors. Finally, to achieve our goal of improving the state of the art in traceback systems, we must address the issues left unsolved by existing techniques and develop solutions with them in mind that are compatible with the various model environments.

4 Implementation and Evaluation

In the following Sections, we will introduce and analyze two of the proposed network tracking systems that seem promising: "Caller Identification System in the Internet Environment" by Jung et. al. [15] and "Distributed Tracing of Intruders" by Stuart Staniford-Chen [26]. We will also discuss experiences with the implementation of the two systems, and give a detailed evaluation of both.

The two approaches were chosen because they each represent one type of monitoring class that is feasible when dealing with traceback. Jung et. al. propose a host-based solution, whereas Staniford-Chen has developed a network-based approach. Each approach makes sense and both call for thorough investigation. There are pros and cons with both of them as will become clear in the following sections.

The two selected approaches fit the model environments as well. Jung et. al.'s work would be most useful in environments where the hosts are tightly controlled such as the Closed Model discussed in Section 2. Also, in the Intranet Model, Jung's work might be appropriate but might be more prone to failure as the control of the hosts may be not as complete as in the Closed Model. Staniford-Chen's work would be useful in areas where we have some level of local network control, but no real control of hosts. This makes his work most appealing for the Academic Model. It may also be useful in the Internet Model as a method of detecting user streams entering and then exiting the internet gateway of a subnetwork.

5 Host-based Tracing of Intruders

In this section we discuss our efforts to reimplement the Caller Identification System in the Internet Environment (CISIE)[15] as introduced by Jung, et.al. We begin by describing CISIE's components and functionality. Next, we present our plan for reimplementation of CISIE and discuss the plan's advantages and disadvantages. As a result of our work to reimplement CISIE, we next present several surprising results that make us doubt that CISIE was ever implemented or tested. We conclude the section by describing new operating system features needed for CISIE to work properly and also detail some cryptographic mechanisms that could make a system based on CISIE more resistant to attack.

5.1 Summary of CISIE

CISIE was developed to trace network attackers across multiple possibly compromised hosts. Such a situation where an attacker remotely logs into a host and from there logs into yet another is called an extended connection. The difficulty in tracing extended connections manually is that hosts do not normally store information that can easily be used to correlate an outgoing connection with its corresponding incoming one. CISIE was designed to alert hosts in an extended connection about all previous "hops" in the connection. The hosts could then use this information to augment their audit trails or even for disallowing logins.

5.1.1 Components of CISIE

CISIE is made up of two components, each of which is installed on every host: the Caller Identification Server (CIS) and an Extended TCP Wrapper (ETCPW). The CIS keeps a trace for each user who has remotely logged into the host. A trace is a list of previous hosts (and user identifiers on those hosts) in the user's extended connection. The ETCPW is an extension to Wietse Venema's well-known TCP Wrapper package[29]. When user requests a login service from the host, the ETCPW tells the local CIS to ask the CIS on the host that made the request for a full trace of the user.

5.1.2 The CISIE Protocol

The initial phase of the CISIE protocol is similar to conventional TCP Wrapper configuration that uses the ident protocol[12]. The server's ETCPW signals the local CIS to do a request for information about the incoming service request. The local CIS then makes a request to the remote CIS which returns a trace for the user. This request consists of the TCP ports that along with the host addresses identify the TCP connection. If this were an ident request, the remote host would reply with the user identifier that owns that end of the connection. Instead, the remote CIS responds with the trace information associated with the user. Note that if this is the first hop of an extended connection, the information is basically the same. The local CIS then stores trace for later use.

The second phase of CISIE consists of authenticating the trace received in the initial phase. To do this, the CIS contacts all hosts in the trace except for itself and its predecessor. It asks the CIS on each of the hosts if the user is logged in. If all reply in the affirmative, the trace is assumed to be valid and the user may continue logging into the system. Otherwise, login is denied and an alarm may be raised.

5.2 Reimplementing CISIE

Initially, we had hoped to obtain the source code of CISIE from the authors of the original paper. However, our attempts to contact them failed possibly due to language issues or because the primary author of the paper no longer appears to be at the Seoul National University. Since we were not able to get the source code for CISIE, we have had to reimplement the system solely based on the paper. Here we describe the details of our effort to reimplement CISIE. First we describe our development platform and then we'll give a description of our design. We finish the section with the current status of the implementation.

5.2.1 Development Platform

We chose to implement CISIE for the Linux operating system with kernel version 2.2.12. Using Linux for this type of development simplifies several aspects of the project. First, since Linux is open source, we can examine and even modify kernel source code if necessary. Second, in Linux the interface used by the ident service is provided via the proc file system. This is a virtual file system that a user process can read to query data from the kernel. In other systems, the data must be manually read from kernel memory.

We chose the Java programming language for implementing CISIE. Java gives us a mostly cross-platform code base and convenient network programming primitives. It also makes it easier to write readable code for others to examine in the future.

5.2.2 Our Design

Our design is very similar to that proposed in the paper[15]. We chose to modify TCP Wrappers as little as possible, and to use features of the wrappers to implement much of what was done in the ETCPW. Our CIS component aims to have the nearly the same functionality as that described in the paper.

In examining TCP Wrappers, we found that by configuring it appropriately we could have the wrapper spawn a program of our choice upon a service request. Furthermore, the wrapper could pass the IP addresses of both ends of the connection to the program. Unfortunately, the wrapper did not allow for passing the other information needed for the trace, the port numbers. This means that the only necessary modifications to TCP Wrappers was to add the ability to pass port numbers to other applications. This is a relatively minor change, and we plan to submit a patch to the package's maintainer as it may be useful in other situations.

When a remote login request is received, the wrapper issues a line of command shell code that launches a program of our design called *Wrap* with the address and port information specified on the command line. Wrap then sends this information to the local CIS using a localhost-limited TCP socket thereby requesting a trace. When the trace request is complete, CIS passes the results back to Wrap which then outputs the trace to standard output so that a calling shell script can allow or disallow the connection.

The CIS portion of our design is nearly the same as that in the paper[15]. The CIS, as is Wrap, is written in Java and uses multiple threads of execution to handle simultaneous requests. Our CIS maintains two listening server sockets. One of them is restricted to the local host and implements the communication between Wrap and CIS. The other handles requests from the CIS's on other hosts.

5.2.3 Status of Implementation

For technical reasons described below, our implementation of CISIE is nearly complete, but it does not function correctly. Wrap is complete, and CIS currently attempts to implement phase 1 of the protocol as described above. Phase 2 of the protocol in which the trace is authenticated is nearly implemented. It has been postponed due to problems replicating phase 1. The incompleteness of the work actually yields our primary result which will be described below.

5.3 Results of The Implementation

In our attempts to implement CISIE, we have discovered a major technical difficulty that is not mentioned in the original work. The problem lies in determining how to link incoming and an outgoing TCP connections so that the trace can be assembled. Initially, it appeared that the same mechanism used in ident could be used for this purpose, but it seems that this is not the case. An ident daemon can determine the client user's identifier for a TCP connection because the process that creates the connection usually runs with the user's privileges. This implies that the local CIS can determine the user identifier on the remote host. The problem of determining the local user identifier associated with an incoming connection to the local host is different. Anything run by TCP Wrapper can not know this immediately because the user has not logged on yet. The local end of the connection was also originally connected by inetd, which runs as root, therefore the user identifier associated with the local end of the connection is root.

The result of this is that there is no easy known way to use the available data to correlate the incoming and outgoing trace. This sheds doubt on whether or not Jung, et. al., had an actual working implementation of CISIE. The matching of incoming and outgoing connections in CISIE appears to be a difficult problem and hence would surely have been an issue covered in their paper. In this light, other inconsistencies show up in the paper. One is regarding the method used to authenticate the streams in phase

2. In Section 2.2 of the paper, item 7 suggests that the CIS's authenticate a substantial part of the trace whereas in Section 2.3.3, we see that the CIS's are just verifying that a user identifier "has a process." Finally, the issue of whether CISIE has been implemented is never directly addressed. We might expect implementation details such as operating system and version, programming language, and possibly real-world experiences with the system, but there are none.

5.4 Modifications to CISIE

We had proposed to experiment with modifications to CISIE, but since we have had to reimplement a system that has substantial undocumented or unknown issues, we have not made any such modifications. Below, we roughly describe some possible additions or techniques to first make CISIE work and secondly to secure and enhance CISIE.

5.4.1 System Features Needed

As described above, the operating system (Linux in this case, but Unix-like operating systems in general) do not provide any easy way to match the outgoing connections with the incoming ones. It may be possible to do so with a variety of techniques with varying degrees of accuracy. We foresee future work in determining the best mechanism to match these connections. Possibilities for this include searching through kernel data structures to attempt to make the match, or modification of the kernel such that it denotes all processes belonging to a given user session with a unique identifier that can be used to match the incoming and outgoing connections.

5.4.2 Securing CISIE with Cryptography

The CISIE protocol as initially presented in the paper ignores the security of the CISIE protocol itself. Because of this, a savvy attacker could pretend to be a previous CIS or possibly query the CIS's in the network to track authorized users to either subvert the system or compromise the users' privacy. The end of the paper suggests a public key method to prevent this, but it is flawed. Here we present a number of suggested modifications to the CISIE protocol to help secure it.

The paper's brief section on securing CISIE suggests that public key cryptography be used to verify that the sender of a Caller Path Request is a CIS. Their suggestion is that a replying CIS would encrypt the reply with the recipient's public key. This does make it so that only the appropriate recipient could read the reply, but it ignores replay attacks. It also ignores the overhead involved with a public key infrastructure (PKI).

Our suggestion is that if the expense of a PKI is warranted, we can do a much better job than this. Since most interactions in CISIE are of the form client request and server answer, we could simply have the client encrypt requests with the server's public key and include a random nonce in the message. The server can then respond by including the nonce sent by the client, encrypting with the client's public key, and signing the message with the server's private key. This simple set of modifications should eliminate most forms of replay attack. Further modifications might have the client sign each request as well so that only valid CIS's can request a trace be done. One area of future study for others is to consider modifications that do not require a public key infrastructure thereby allowing for lighter weight protocols.

5.5 Conclusions about CISIE

We have made a substantial effort to reproduce results reported by Jung, et. al. Unfortunately, we have been unsuccessful and even doubt that an implementation existed prior to the writing of their paper. We have also discussed modifications to the Linux (or other) operating system to make CISIE work. Finally, cryptographic methods have been considered to help secure the CISIE protocol, but implementation seems unwarranted until the protocol has been shown to work.

6 Network-based Tracing of Intruders

In order to determine the identity of a network attacker, it may be necessary to be able to trace back a chain of connections between network hosts to its origin. One might consider performing this task at the host level of each machine in the chain. However, there are a number of problems with this. A user might have multiple open connections, making it difficult to associate specific ones to those on other hosts. Header information can change, also complicating the successful matching. A user might even change services within the chain of connections on a particular host, e.g., from *telnet* to *rlogin*. Covert channels on a host, even though unlikely, can render a host-based solution virtually worthless. Last but not least, a host can be compromised by an attacker, leaving the usefulness of any audit data questionable. Thus it seems reasonable to also examine and pursue network-based approaches.

One approach, taken by Stuart Staniford-Chen [26], is to analyze properties of the TCP stream itself between pairs of hosts. The goal of the approach is to find certain characteristics within the stream that uniquely sets it apart from other streams. Ideally, those characteristics will persist throughout the chain of connections. Staniford-Chen introduces the concept of *thumbprints* to achieve this goal. A thumbprint is a short piece of data that summarizes the content of a connection. It should have the following properties: small size, sensitivity to the content, and robustness. Furthermore, the thumbprints should be additive, easy to compute, and be easily comparable. Under the current model, only *telnet* and *rlogin* sessions are considered. This is due to the fact that content within the stream is used, and these services do not change the content at each hop as some encrypted services do.

In this model, a number of network sniffers are placed at appropriate points in the network, and at each location information about each TCP stream is collected. Each stream is divided into intervals, typically of one minute length. This allows for comparing portions of the stream. Over the time of one interval, a vector, represented as an array of integers, is used to keep track of the number of occurrences for each of the possible 128 ASCII characters in the TCP packets of the stream. This vector by itself might suffice as a thumbprint as it identifies a connection or a stream. However, the vectors are quite large, and effectively comparing two vectors might become difficult.

To compress the size of the thumbprint and to provide better means of comparing them, Staniford-Chen proposes principal component analysis to provide a function that emphasizes more unusual characters over those that are frequently used in all or most telnet or rlogin traffic.

6.1 Local Thumbprints

The author proposes what he calls *local thumbprints*, where "[...] the thumbprint is a sum of terms, each of which depends only locally on the character stream to be thumbprinted" [26]. Thus, for a simple thumbprint vector T_j with j components, he suggests $T_j = \frac{1}{n}\sum_i \phi_j(a_i)$, where n denotes the number of characters in the alphabet and the a_i stand for those individual characters. In the experiments described in the paper, this definition of a thumbprint is used, and therefore will be the only one discussed here.

The function ϕ_j is obtained through principal component analysis described below. Essentially, for each character a_i of the alphabet and each component j of the thumbprint, ϕ_j is simply a factor that helps emphasizing characters that are more atypical for a telnet session. A big advantage of having a fairly easy thumbprint function is that it is very easy to compute. Most of the work is spent obtaining the function, which can then be stored in a simple look-up table.

6.2 Comparing Thumbprints

For comparing thumbprints effectively, Staniford-Chen develops a scheme that magnifies differences between components and considers the proposition that, when successive thumbprints match over a longer time period, they are likely to be related. For this purpose, a probability approximation $P'(\delta)$ is created by randomly choosing pairs of thumbprints out of a sample pool and calculating logarithm of the product of the differences of the individual components. The resulting distribution can then be used to approximate the probability of a given δ. The smaller the probability, the more likely it is that the streams are related. For successive intervals of time, the individual probabilities can be multiplied. The author refines this method to a great extend, using statistical models, which are beyond the scope of this report.

Again, as for the ϕ-function discussed above, most of the work is performed in advance on sample data. Once the approximate distribution is found, comparing pairs of thumbprints is a matter of a few multiplications and additions. Note, however, that one cannot know in advance which thumbprints should be compared with each other. The author doesn't address this issue directly.

6.3 Principal Component Analysis

Principal component analysis is a methodology adapted from statistics, and Staniford-Chen gives two references [16, 9]. The goal is to find directions in a set of vectors that respresent more of the variation within those vectors than any other direction. For this purpose, a covariance matrix is calculated over the elements of the vectors. Then, the eigenvalues and eigenvectors of the matrix are computed. The eigenvectors that belong to the largest eigenvalues are then taken, and its individual components describe the function ϕ.

Assume that the thumbprints have c components over an alphabet of n characters. For a sample of N vectors, the following steps need to be taken to calculate ϕ:

1. Calculate the covariance matrix $C_{jk} = \frac{1}{N}\sum_{i=1}^{N}(x_{ij} - \overline{x_j})(x_{ik} - \overline{x_k})$, where $\overline{x_j}$ and $\overline{x_k}$ are the averages in each direction, respectively.

2. Take the c largest eigenvalues of $C_j k$ and the eigenvectors that belong to those, $\vec{e_1}, \cdots, \vec{e_c}$.

3. ϕ_1 now corresponds to $\vec{e_1}$, ϕ_2 to $\vec{e_2}$, and so on. I.e., $\phi_j(a_1)$ is the first component of $\vec{e_j}$, $\phi_j(a_2)$ is the second component of $\vec{e_j}$ etc.

6.4 Implementation

We asked Mr. Staniford-Chen as well as faculty at the University of California Davis for source code. This request being denied, the thumbprinting approach is currently being implemented using the machines provided for the project. The software platform for the project is Linux, Kernel version 2.2.12. The following assumptions had to be made, as the paper does not discuss them:

- All TCP packets without explicit data content such as SYNs, ACKs, and FINs are not considered for the thumbprint calculation.
- It is assumed, that the eigenvectors obtained as described above indeed constitute the factors of the function *phi*. The author never explicitly mentions this, but it is the only reasonable approach.
- It is assumed that taking the first c largest eigenvalue/eigenvector pairs also results in the thumbprint consisting of c components.
- TCP connections will all start with a SYN packet and will all end with a FIN packet.

Our implementation differs from that of Staniford-Chen in that we differentiate between the two ends of a TCP connection. By doing so, we hope to achieve an even better characterization of the stream. If desired, one can always add the two thumbprints to achieve one that characterizes both directions of the stream.

For the purpose of examining the TCP streams, the *libpcap* library [17] was utilized. Much of the programming code was adapted from Stevens [28] from examples using that library.

The parameters were kept the way as described in the paper [26]. Thus the time interval has a length of one minute, only the first 128 ASCII characters are considered, and the thumbprints will consist of 6 components.

The program establishes a counting array for each individual connection, that is, for each 4-tuple <*source IP-address, destination IP-address, source port, destination port*>, an array with 128 counters - one for each possible character - is created. As data is being sent, the counters are increased appropriately. An alarm will interrupt every 60 seconds, and the arrays can then be treated as the vector that describes the stream. The thumbprint function can then be applied and the thumbprint be stored. After that, the counters in the array are reset to 0.

Obtaining good quality sample data is a very important aspect of the implementation. Indeed, the overall success will almost solely depend on it. After a collection of artificially generated sample TCP traffic data that was provided to us early in the project proved to be insufficient for our purpose, actual user data is currently being gathered within the Computer Science Department of Purdue University.

For this reason, the developed program yet lacks a proper thumbprint function, and the comparison routine is not yet existent.

By the time this report is finished, the first data set should be available, and we will be able to proceed to evaluate the ϕ-function as well as the probability distribution $P'(\delta)$. A second data set will be generated and compared against the first. After that, the ϕ-function can be incorporated into the program, and the comparison routine can be developed.

6.5 Shortcomings

There are several flaws and shortcomings with the thumbprint approach.

1. The whole computation of the thumbprints is solely based on content. If encryption is used with different key pairs between different hosts, the thumbprints will differ greatly, indicating (falsely) different TCP streams.

2. It is unclear why Staniford-Chen limits his work to an analysis of content. Clearly, similar vectors to the content-based ones can be created for packet frequency and idle times. Those could be processed in a similar fashion and evaluated. Thumbprints obtained that way could then be compared within their own classes (content, timing, idle-time), or one could think of a way to combine them, taking all the characteristics of the stream into account. This way, the thumbprinting scheme would not solely depend on content alone, and maybe could still suffice in analyzing encrypted traffic. However, there are also ways to perturb the other two characteristics of the stream.

3. The choice of Principal Component Analysis to determine the optimal thumbprint function is disputable. Even though "[the] aim of principal component analysis is to take a series of vectors and find a set of linear combinations of the components which explains the maximal proportion of the variance of the vectors [...]" [26], the author never argues why this approach is the best choice. Other methods for obtaining ϕ, such as a genetic algorithm or machine learning scheme also seem feasible.

4. Several other choices the author made seem rather arbitrary and are not explained satisfactorily: why sample the stream in intervals of one minute length? He only differentiates against a 10 second long interval. Why use only the first 6 principal components?

5. The author doesn't provide any ideas as to how to choose thumbprints that should be compared with each other. Initially, when comparing a particular thumbprint of a stream that was used for an attack on host A to thumbprints of host B, one might limit the range of prints to compare to those that lie in a certain time interval prior to the attack. However, if no thumbprints match, it does not guarantee that host B was not part of the chain of connections, as an attacker could certainly have caused a delay in execution of commands. Thus, in the worst case, all thumbprints of host B have to be compared, resulting in a very negative performance impact.

6.6 Conclusions about Staniford-Chen

Stuart Staniford-Chen's work is a start. It allows for an efficient and fast computation of local thumbprints for *telnet* and *rlogin* sessions. The underlying theory has a well-established foundation in statistics. A major advantage of the approach lies in the fact that a vast part of the computation can be performed in advance, reducing the actual calculation of a thumbprint to a few mathematical operations. Local thumbprints seem to work extraordinarily well within the given environment, even though this claim has yet to be verified.

However, there are many unanswered questions. The most obvious flaw is, of course, the solitary focus on data content alone. Staniford-Chen makes it rather easy for himself this way, leaving out other important information about the data stream. Besides content, there is also meta information about the stream. One can look at timing, idle times, and at header information that remains constant. Ideally, all those factors should be incorporated into a thumbprint. In order to prevent a thumbprint from becoming useless if one or more of the session characteristics are circumvented, the components should be stored separately. A new comparison function needs to be created that combines the components, but also allows for individual inclusion or exclusion of features.

Even though Staniford-Chen gives a thorough discussion about the underlying theory of his approach, it becomes obvious that the work he has performed covers only a small part of what needs to be done. In many cases, he arbitrarily picks the value of a parameter, sometimes after only a little pre-evaluation, and then sticks with it for the rest of his experimental results. There are many parameters to this problem, generating a huge search space for the optimal combination. There are many open questions concerning these parameters. What is the best number of thumbprint components? How long should the timing interval be? Are local thumbprints as good, better, or worse than the higher-order thumbprints he discusses? What constitutes the best thumbprint function? All these questions will have to be answered in the future.

Another concern is that of the thumbprint ϕ-function. The way it is constructed leads directly to the following questions: What kind of sample data needs one to take to best capture "ordinary" *telnet* and *rlogin* features? To what extent do individual or sequences of characters stand out to characterize individual streams? Is there a single "ideal" thumbprint function? Right now, the ϕ-function is static. It is pre-computed, and will never change from that point on. It might be desirable, to be able to adjust the function, as features of TCP streams might change over time. As a matter of fact, this might be a major disadvantage of principal component analysis. In order to change the function, a new covariance matrix needs to be computed, and its eigenvectors and eigenvalues calculated. This amounts to the same extensive work as the initial computations. A machine learning approach could lead to more efficient results. Once the training phase with the sample data set is complete, the functions can be updated by incorporating new samples into the process. This can be done without having to recompute any previous calculations. Thus, a machine learning approach and its effectiveness might be well worth pursuing. This way, it might also be possible to find other important distinctions between data streams than the greatest variation.

Comparing TCP streams on a network level is very important. Staniford-Chen introduces a new concept for doing so, and he explores a small fraction of the overall framework. There is yet plenty of research to be done in the area. If the obstacles mentioned in this report can be overcome, this technique could evolve into a very useful methodology. Note, however, that any attempt of matching streams is subject to countermeasures by a determined attacker. In addition to encryption, tools can be created that randomly obscure timing and idle time information on a per session basis. Streams can be split up, sent through chains of different hosts, and later be combined again. Those are problems that are yet to be solved.

7 Conclusions Regarding Our Implementations

In this paper, we have discussed our work towards implementation and evaluation of two known approaches to tracing network intruders. CISIE appears to have been designed but possibly not implemeted by its authors. Our attempts to recreate their system uncovered the crucial problem of correlating incoming TCP connections with the outgoing ones. This is difficult although probably doable on the host by either using intimate knowledge of the operating system to make the correlation or by making minor modifications to the kernel of the operating system itself. We have also outlined some possible approaches to securing CISIE against tampering. We have also discussed Staniford-Chen's thumprinting techniques. We conclude that while his work is a good start in the area, there remains a great many unanswered questions about the work. There are many parameters of his approach that seem set arbitrarily, and it is unclear that the scheme will work in the real world, in which attackers use encryption to hide the contents of their data streams.

8 Attack Traceback Workshop

The Attack Traceback Workshop was held by CERIAS from September 6th through the 8th. For the convenience of the those invited, the workshop was held at the Sheraton Gateway Suites in Rosemont, Illinois, near Chicago's O'Hare International Airport.

The workshop's goals were to learn the current work being done in the area of attack traceback, share ideas about the topic, and generate diverse discussion between various parts of the community. To accomplish this, we invited academic and industry leaders who are known to have done research in attack traceback, business representatives in networking and software, and legal experts from both the private and public sector.

The schedule of the workshop was laid out as follows. The workshop began with an informal reception held on the evening of September 6th. This encouraged attendees to arrive early and allowed them to get to know one another. The first half of the 7th was occupied by introductory presentations by the attendees about their own work. The remainder of the 7th and early portion of the 8th was devoted to group meetings as detailed below. The workshop concluded around noon on the 8th with open discussion on the outcomes of the workshop.

8.1 Presentations

Introductory presentations were initially given by Eugene Spafford, Clay Shields, and Tom Daniels, all of CERIAS. Following these presentations, others were allowed to speak about their own work and perspectives. Below, we briefly summarize the content of these presentations.

Eugene H. Spafford, CERIAS, Purdue Dr. Spafford began the workshop with some opening remarks and thanked the attendees for their time. He also laid out the agenda for the workshop.

Clay Shields, CERIAS, Purdue Dr. Shields gave brief descriptions of the problems that lead to the need for attack traceback. These are as described earlier in this document.

Tom Daniels, CERIAS, Purdue Mr. Daniels pointed out that there is a more general problem than attack traceback, namely tracing network traffic. He also outlined eight desirable properties of traceback systems and possible tradeoffs. They are accuracy, precision, subversion resistance, low overhead, low cost, scalability, realtime, and privacy.

Adrian Perrig, UC Berkeley

Routers probabilistically mark packets (5%) in 16 bits of the IP header (fragmentation information). A hashing scheme is used that was not further discussed in detail. In a basic version, the neighbouring routers' IP addresses are stored. In a more advanced scheme, the 16 bits are split in a 5 bit distance and 11 bit hash of the routers' addresses. This implies the existence of an approximate topology map of upstream routers that is accessible to a victim. Best results were achieved with a 3 bit flag id, a 5 bit distance and an 8 bit hash of the addresses

TESLA

Routers commit to a chain of keys which they use to mark packets. The chain is released after a delay so that markings can be verified by clients.

Steven Bellovin, AT&T Labs
ICMP traceback messages

The methodology requires a large amount of traffic to be generated. With a low probability (1/10000), routers send ICMP packets with information about the packet to the destination. The TTL is set to 255, so that a distance measure can be established.

Congestion controls

If data streams are ignoring control flow messages from the routers, they are flagged as misbehaving. A penalty box will discard more packets from misbehaving streams in order to balance the load more evenly.

Stuart Staniford

The goal is to automate the process of obtaining ISP logs accross (international) boundaries. The creation of an agency was suggested that locally allows to query ISPs that fall under its jurisdiction. If lines of jurisdiction are crossed, the agency has to contact its proper peer to obtain the necessary information. Law enforcement need only contact the agency, which will perform the traceback without disclosing any sensitive other information.

Dan Sterne, NAI
Cooperative traceback and response

For traceback and other purposes, an intrusion detection and isolation protocol was devised. If an event occurs, a query is being propagated among cooperating neighbours to the source of the attack or to an uncooperative network. The approach requires a controller that monitors for intrusions and then handles the queries and propagets them to its peers.

Felix Wu, UC Davis

While the traditional firewall technology separates the network vertically into protected zone and public network, the aggregated internet traffic toward a particular destination can also be separated horizontally into different packet flows. Horizontal separation by itself is not a new idea as it has been realized implicitly or explicitly in many existing network protocols and architecture. However, this important concept has not been integrated into today's IDS/IRS technology to handle many difficult network security issues such as denial of service attacks. The concept of Horizonal Separation is utilized to resolve the problem of DDoS (Distributed Denial of Service) attacks in an inter-domain environment. Based on this concept, a prototype system called DECIDUOUS (DECentralized IDentification of intrUsion sOUrseS) was developed.

IPSec tunnels are used to forward packets into other networks. Thus endpoints at borders can be easily identified. It can be shown that two permanently established tunnels per borderpoint are sufficient for routing that allows such a traceback.

John Lynch, Department of Justice Mr. Lynch outlined some general areas of importance to the DOJ regarding traceback information. He was concerned with how data is collected, chain of custody issues, and types of

data collected. He pointed out that information passed along as part of a protocol is better than protocols that query back to remote sites. This is because it could complicate the evidentiary requirements as multiple representatives might be needed to authenticate the data. He continued
William Cook, Winston and Strawn
Abigail Abraham, Assistant State's Attorney, Cooke Co., Illinois

8.2 Group Meetings

Following the presentations of various members of the legal communitee, the conference divided into three discussion groups to discuss the following topics:

- Group 1 – Business and Social Aspects of Traceback
- Group 2 – Technical and Operational Issues involved in Traceback
- Group 3 – Legal, Regulatory, and Marketplace Issues

The next few sections describe the results and conclusions of each of these working groups.

8.2.1 Business and Social Group

The business group identified the following major problems that businesses generally face with attacks:

- It is very hard to determine who is liable if an incident happens.
- Where does an intrusion originate?
- How can one distinguish between legitimate requests by clients and competitors illegally gathering information? What is the line of tolerance for a business?
- Is the business being used as a third party for an attack?

In the discussion, it became clear that different businesses have different expectations. A business that operates internationally will need clear legal guidelines in forms of not only laws, but also international treaties that signify cooperation among those countries but also lay out the rules as to what a business has to consider when employing traceback technology.

A small business that operates within its national boundaries might be more concerned with the cost of employing the technology and what implications it would have for them if they failed to take the required steps necessary to adhere.

In general, the question arises what happens if the requires cooperation is not achieved. Data havens on an international scope or businesses refusing to cooperate can seriously hinder traceback efforts to a point where it will become useless for a business as the above described problems can no longer be solved.

Furthermore, currently traceback technology can only provide a network address as the final source of an attack. From a business perspective, this might not be sufficient. Cases can be thought of where the network address alone gives no clues whatsoever as to who the human being behind the attack is. At that point, agin, the liability question arises.

From a technology perspective, businesses are very concerned about the misuse of traceback: is it possible to deceive or even frame people or businesses with the traceback technology? This will be one of the driving aspects concerning the acceptance of traceback.

Another crucial point is the question who will control traceback. In general, it is very unlikely that there will be an overall acceptance as to who can be trusted fully by everyone and therefore should control it. It might be worthwhile trying to establish some sort of trust relationship models for this purpose, maybe in form of a grammar.

In conclusion, the members of the discussion group agreed that businesses will need some sort of incentive in order to agree to traceback. Money and cost being the most prominent drivers, the necessary incentives could be achieved in various ways or some combination thereof:

- Offer economic rewards. This could be done by businesses themselves, invoking penalties on uncooperative businesses, or intensifying economic relationships with coorperating ones.

- Establish clear legal guidelines and liability framework, so that costs of litigation become tangible.

- Let insurance companies gather statistical data which will then be used to establish an insurance model for e-commerce. Then the insurance premiums can be used as a cost measurement.

8.2.2 Technical and Operational Group

Initially, the technical and operational group began by struggling with defining the problems in network traceback. Eventually, the group decided to build an operational model of the problems facing someone trying to trace an attack. Shown below, the model describes a network scenario where three different types of intermediaries are being used to obfuscate the source of the attack.

From the victim's perspective, each number labels a different research problem as listed below.

1. Determining the source of a packet

2. Determining the source of the stimulus of a reflector

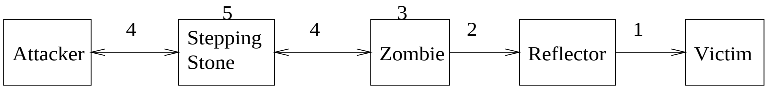

Figure 5: Operational Model

3. Reversing 2-way to 1-way application-level laundering

4. Determining the source of a 2-way connection

5. The stepping stone problem: 2-way to 2-way connection laundering

Related work was done on terminology. A zombie would represent a host that may be compromised that may be running a trojan horse. The time between its stimulus and its output may be arbitrary. A reflector is bound to its behavior by a given protocol specification and is not considered compromised, but behaving normally. A stepping stone may be compromised, but it is being used to relay data in a time-bounded fashion such as a host being used to launder user session streams.

The discussion next considered tracing causality across a host. If we place a limit on the delay, what is possible? It was agreed that many things may be possible in some environments, but it was unknown what all of them were. This conversation devolved into the secure OS problem.

The group concluded that basic models and terminology are still needed, but we made a bit of progress. There are two main problems: packet tracing in an understood network and causality within the host. It was agreed that packet tracing is possible given sufficient driving force, but it was unclear how much is possible in arbitrary hosts.

In the final session, the group worked on identifying small steps that can be taken by the community. These were grouped as follows:

- Nomenclature

- Identifying Source of Packets More work is needed on the following in routers: better user interfaces, diagnostic features, querying features for tracing packets, marking facilities, logging and storage features, and more flexible control interfaces. Also, the existing techniques for tracing packets must continue to be improved and be better understood.

- Stream Matching Using content may be legally difficult. Further work must be done to advance these techniques.

- Determining Causality Through Hosts It's important to determine the limits (theoretical and practical) both on the host and outside of it.

- Limits of overall approaches Further work can address simulating and modeling proposed techniques. Interesting measures such as what is the cost of providing a given level of accuracy should be investigated.

- Infrastructure Management New work should address how to best manage a distributed tracing infrastructure. This would include work on trust models, cooperation, deployment, and where to maintain trace state.

8.2.3 Legal Group

The legal group spent the first part of their time discussing the various applicable statutes and laws in the United States of America. Title 18 of the United States code has several relevant sections:

- **1030** – Computer Hacking
- **2511-2521** – The Wiretap Act
- **2701-2711** – Disclosure of information to government
- **3121-3126** – Pen Registers & Trap and Trace

The government is bound by stricter rules than corporations when it comes to data sharing for activities such as network traceback. Sharing of some of the network data between companies is permissable, but the sharing of the same data with anyone acting as an agent of the government is illegal. It was noted that this twist was likely the result of business lobbying for an exemption to allow them to share marketing information.

It was the consensus of the individuals participating in the discussion that the disclosure of header and routing information to the government would be legal, but disclosing any of the content would not. However, it was difficult to come to any consensus as to what the definition of "header information" really described. Certainly the source and destination information on a packet is header and routing information; however, if the payload of the packet is an SMTP header describing where email is being sent/routed is that content (and illegal to use) or header (and legal)? It may take a few actual court cases to get things sorted out.

Another twist that ISPs would be faced with is that the release of content by a legitimate user would likely be illegal (barring any contract negotiated in advance between the ISP and the customer), but an non-legitamate user would not be protected in this fashion. Of course, the tricky part is determining the legitimacy of any particular connection.

The lawyers in the group described an interesting situation that occurs regularly in the world of voicemail and now has been applied to electronic mail as well. Since the government is barred from content of data in transit, an unheard voicemail or unread email is off limits to them without a warrant. However, once it has been read, it is considered to be just a file on a computer system that could be shared as any other file was.

In regards to the technical presentations made earlier in the day, the lawyers indicated that making queries to block traffic (e.g. Distributed Denial of Service attacks) would be acceptable, but the same type of queries to be used to trace back a connection would be unacceptable and possibly illegal.

9 Overall Conclusions

We have presented an overview of our work in attack traceback. As part of this work, we have demonstrated the considerable difficulty we have had in recreating the past work of others and discussed several approaches to dealing with these difficulties. Also, we have summarized the events of the Attack Traceback Workshop in their business, legal and academic aspects. The greatest conclusion to be drawn from all of this is that much is left to be done in this area. There are many approaches being defined to solve the problem, but we still do not know how they compare and what their ramifications are likely to be.

References

[1] Assistant, L. P. W. Available at `http://www.bell-labs.com/projects/lpwa`.

[2] Bellovin, S. M. Security Problems in the TCP-IP Protocol Suite. *Computer Communications Review 19*, 2 (April 1989), 32–48.

[3] CA-96.21, C. A. TCP SYN Flooding and IP Spoofing Attacks. http://www.cert.org/advisories/CA-96.21.tcp_syn_flooding.html, September 1996.

[4] CA-97.27, C. A. FTP bounce. `http://www.cert.org/advisories/CA-97.27.FTP_bounce.html`, Dec 1997.

[5] CA-97.28, C. A. IP Denial-of-Service Attacks. http://www.cert.org/advisories/CA-97.28.Teardrop_Land.html, December 1997.

[6] CA-98.01, C. A. 'Smurf' IP Denial-of-Service Attacks. http://www.cert.org/advisories/CA-98.01.smurf.html, January 1998.

[7] CA-98.13, C. A. Vulnerability in Certain TCP/IP Implementations. http://www.cert.org/advisories/CA-98-13-tcp-denial-of-service.html, December 1998.

[8] Chang, H., and D.Drew. DoSTracker. This was a publically available PERL script that attmepted 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.

[9] Chatfield, C., and Collins, A. *Introduction to Multivariate Analysis.* Chapman and Hall, London, 1980.

[10] Chaum, D. L. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM 24*, 2 (February 1981), 84–88.

[11] Clark, D. The Design Philosphy of the DARPA Internet Protocols. In *Proc. ACM SIGCOMM* (August 1988), pp. 106–114.

[12] Johns, M. S. Identification protocol. Request for Comments 1413, February 1993.

[13] Joncheray, L. Simple Active Attack Against TCP. In *Proceedings of the Fifth USENIX UNIX Security Symposium* (Salt Lake City, Utah, June 1995).

[14] Jung, H. T., Kim, H. L., Seo, Y. M., Choe, G., Min, S. L., Kim, C. S., and Koh, K. Caller id system in the internet environment. In *UNIX Security Symposium IV Proceedings* (1993), pp. 69–78.

[15] Jung, H. T., Kim, H. L., Seo, Y. M., Choe, G., Min, S. L., Kim, C. S., and Koh, K. Caller identification system in the internet environment. In *UNIX Security Symposium IV Proceedings* (1993), pp. 69–78.

[16] Krzanowski, W. *Principles of Multivariate Analysis.* Clarendon Press, Oxford, 1988.

[17] Laboratory, L. B. N. ftp://ftp.ee.lbl.gov.

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

[19] Morris, R. A Weakness in the 4.2BSD Unix TCP-IP Software. Tech. Rep. 17, AT&T Bell Laboratories, 1985. Computing Science Technical Report.

[20] Ptacek, T. H., and Newsham, T. N. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Tech. rep., Secure Networks, Inc., Suite 330, 1201 5th Street S.W, Calgary, Alberta, Canada, T2R-0Y6, Jan. 1998.

[21] Reed, M. G., Syverson, P. F., and Goldschlag, D. M. Proxies for anonymous routing. In *12th Annual Computer Security Applications Conference* (December 1995), IEEE, pp. 95–104.

[22] Reiter, M. K., and Rubin, A. D. Crowds: Anonymity for web transactions. Tech. Rep. 97–15, DIMACS, April 1997.

[23] Rowe, J. Intrusion detection and isolation protocol: Automated response to attacks. Presentation at RAID'99, Sep 1999.

[24] Staniford-Chen, S., and Heberlein, L. Holding Intruders Accountable on the Internet. In *Proc. of the 1995 IEEE Symposium on Security and Privacy* (Oakland, CA, May 1995), pp. 39–49.

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

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

[27] Stevens, W. R. *TCP/IP Illustrated Volume 1.* Addison-Wesley Publishing Company, 1994.

[28] Stevens, W. R. *UNIX Network Programming*, second ed., vol. 1. Prentice Hall, Upper Saddle River, 1998.

[29] Venema, W. TCP wrappers. available at `ftp://ftp.win.tue.nl/pub/security/tcp_wrappers_7.6.tar.gz`.

[30] web site., A. Available at `http://www.anonymizer.com`.

[31] Zhang, Y., and Paxson, V. Stepping Stone Detection. Presentation at SIGCOMM'99, New Areas of Research, August 1999.