PIONEERING TECHNOLOGIES FOR A BETTER INTERNET

Cs3, Inc.

5777 W. Century Blvd.
Suite 1185
Los Angeles, CA 90045-5600

Phone: 310-337-3013
Fax: 310-337-3012
E-mail: info@cs3-inc.com

Survey/Analysis of Levels I, II, and III Attack Attribution Techniques[1]

Sponsored by:

Cs3, Inc.
5777 W Century Blvd
Suite 1185
Los Angeles, CA 90045
(310) 337-3013

PROJECT CONTACT: K. Narayanaswamy
EMAIL: swamy@cs3-inc.com
REPORT DATE: 27 April 2004

[1] This work was supported by Advanced Research Development Activity (ARDA), under contract NBCHC030115. The views expressed in this document are those of the author(s) and CS3 Inc., and not those of ARDA or its representatives.

Table of Contents

Introduction

Cs3 is currently executing a research project for ARDA entitled “Attack Attribution Techniques for Hybrid, Cooperative and Non-Cooperative Infrastructure”. This project was proposed in response to an ARDA solicitation [BAA 03-03-FH], specifically the area of attack attribution. The BAA defined the general attribution problem into four different “levels”:

1. to the specific hosts involved in the attack;
2. to the primary controlling host;
3. to the actual human actor;
4. to a higher organization with a specific purpose to the attack.

The BAA specifically indicates interest in techniques for attribution in situations where not all of the network infrastructure cooperates in the attribution effort.

This document is an early deliverable that surveys and assesses the different existing techniques for Level 1, 2, and 3 attribution. The results of this assessment highlight shortcomings of existing approaches that will be addressed during the course of the research.

Chapter I

Techniques for Level 1 Attack Attribution

I.1 Problem Statement

Our model is that an IP packet, P, is generated by a machine, G, forwarded by a sequence of IP routers, and finally, if not dropped along the way, delivered to a recipient machine.

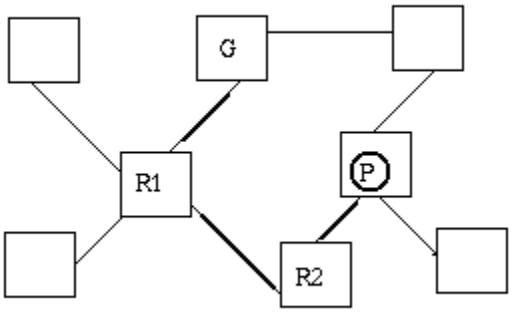

Figure I.1. **Diagram of Typical Network**

Figure I.1. illustrates packet P in the middle of its journey after having been generated at G and forwarded along the darkened links by routers R1 and R2. The goal of Level 1 attribution is, given P, to identify G. This identification might, in general, be requested by the recipient, any of the forwarding routers, or any other machine to which those routers communicate information about P. Note that every IP packet is supposed to contain the IP address of the machine that generated it in the source address field. If this requirement were enforced then Level 1 attribution would be trivial.

The rest of this section is devoted to clarifications of the problem statement above.

I.1.1 General Terminology

Level 1 attribution is often called **traceback** in the literature, and we will use that term to mean the same thing as level 1 attribution. Similarly, **tracing** a packet is used to mean finding the machine that generated it. We use the term **tracker** to mean the party attempting to carry out traceback. The machine that generated P is called the **origin** of P. The **source address** of P is the IP address contained in P that is meant to be used as a return address. This *should* be the IP address of the origin of P. The sequence of machines that sent P is called the **forwarding path** of P. This sequence begins with G. Normally the forwarding path has no loops, and in that case one machine on the path must occur either earlier (closer to G) or later than another. The earlier machine is said to be **upstream** of the later machine, and the later machine is said to be **downstream** of the earlier one.

The BAA refers to identification of “hosts”, which we think of as computers that are primarily used by single human users. The machines that generate packets do not necessarily fit that

description. We generally avoid the term "host". We use the term **machine** to cover hosts, routers and any other special purpose devices that receive or send network traffic. Level 1 attribution is supposed to identify the "machine" that generated the packet, whether that machine is a host or not.

I.1.2 What Is an IP Router and What Do "Forward" and "Generate" Mean?

By IP **router**, we mean a machine that acts substantially in accordance with [rfc1812]. That RFC describes forwarding, in particular how packets received by a router are transformed into packets that it subsequently sends. The term **forward** is used only for the transformation, described in RFC 1812, that affects only a few fields of the IP header. It does not apply to such things as replies to packets addressed to the router or to ICMP replies that the router sends to source addresses of packets it cannot forward. These are considered to be "generated" by the router. Similarly NAT devices that alter source addresses are considered to generate the packets they send with altered source addresses. Although it is important to find the packets that cause these packets to be generated, that problem is classified as Level 2 attribution, which is to be addressed elsewhere.

A packet that is sent by a machine but not forwarded in the sense above is considered to have been **generated** by the sending machine.

I.1.3 Why Only IP?

In general, trackers are interested in more than IP, e.g., tracing a non-IP packet, or tracing an IP packet to a non-IP device. We limit the discussion to IP because:

- The vast majority of actual (at least non-local) traffic to be traced is expected to be IP
- Almost all of the work to date has concentrated on IP
- The methods applicable to IP and to non-IP traffic overlap substantially

I.1.4 What Does it Mean to "Identify" a Machine?

Ideally the tracker would like to find the physical device that generated the packet. In practice he is generally satisfied with an IP address that is unique within the network visible to him. In some cases, the generating machine may have no such address. In this case, the best result possible would be the address of the first forwarding router that does have such an address, which serves as approximate identification. Any further information, such as the fact that the packet arrived at that router from a particular wire, improves the approximation. This kind of attribution is generally the best that can be achieved with the techniques we review here, and even such results are possible only in the presence of very high levels of cooperation.

In the general case, lower levels of cooperation will generate less complete traceback data, restricting the origin to a set of possible IP addresses. The quality of the result is then related to the size of that set, with smaller sets considered higher quality.

One area that we do not address here is characterization of physical signals. These can be very useful for identifying a machine that is one hop away, but for the most part we are concerned here with machines that are more than one hop away.

I.2 Evaluation Criteria for Traceback Methods

Section I.3 is devoted to descriptions of how extant techniques work, how they can be attacked and what defenses could be used against those attacks. For purposes of comparison, section I.3 also evaluates the techniques with respect to the criteria presented in this section. The motivation for each criterion is described with that criterion, but in general the criteria are considered to be relevant to the task of traceback within the IC context.

The criteria are presented in the form of yes or no questions that will be answered for each method in section I.3. Each question has a preferred answer, meaning that a method with that answer has an advantage over a method with the opposite answer. The answers in section I.3 include plus or minus signs to indicate whether they are desirable or undesirable. Section I.3 does *not* compare methods in terms of how "useful" or "powerful" they are, or how the power of the method depends on the number or location of cooperating machines. These subjects are discussed [Cs3 ARDA 1].

I.2.1 Single Packet

Does the mechanism allow traceback of a single packet?

Some methods apply only to large sets of packets presumed to come from the same place. These methods are generally designed to trace floods. "Yes" means that after the arrival of a single packet, traceback data can be found for that individual packet.

Note: For purposes of the intelligence community we believe that it is necessary to trace single packets.

I.2.2 Existing Commercial Routers

Does the method work with existing commercial routers?

Some methods require changing the way routers work. These are already at a disadvantage compared to those that do not. The magnitude of the disadvantage is proportional to the difficulty of the changes required. If no changes are required or the changes can be accomplished by changing the configuration of the routers already in the network then the answer to this question is "yes". What can be done by reconfiguration actually depends on which router is used, but we answer yes if we expect that most commercial routers can be reconfigured to do what is needed.

If the answer is "no", the difficulty depends partly on what changes in router behavior are required by the method in question, and partly on the bandwidth that must be handled by the routers to be modified. We see three general cases, which we now present along with the best solution we see for each case.

If the router in question has to deal with data rates less than about 1Gbit it is relatively easy to build a custom router out of cheap standard hardware running open source software. This software can be modified to do whatever is required (within reason) by the method in question.

If the method requires special processing (outside of what a normal commercial router could do) for only a small minority of the packets that it receives, a high speed commercial router could be configured to identify the packets that need special processing and forward those to a separate lower speed but more programmable machine. That machine would do the specialized processing, and then return the (possibly modified) packets to be forwarded on. We will mention this alternative below as applicable.

In the worst case, specialized processing is needed at high speed. There is currently no solution to this problem other than for the few organizations able to design and build high-speed routers. We do have a solution in mind for this.[2]

I.2.3 Advance Notice

Does the method require advance notice of packets to be traced?

Some methods are too expensive to be used for all packets all the time. For example the method might slow down or even disrupt normal traffic while in operation. In that case, the tracker would have to start the data collection when he expected packets that he would want to trace. As another example, a monitor with finite memory must limit the set of packets it remembers. One solution to this problem is to remember only recent packets. However, such a solution limits traceback to recent packets. The monitor can remember packets for a longer time if it need remember only a small subset that is considered interesting. In that case, the tracker has to tell the monitor which is interesting, and that must be done before those packets can be recorded. Requiring advanced notice is a fatal flaw for cases where the tracker only discovers after a packet arrives that its origin is of interest.

I.2.4 Additional Communication

Does the method require additional communication after identification of packets to be traced?

Typically, methods that rely on memory at cooperating machines require communication with those machines in order to query them on what they have recorded and receive an answer. This is a disadvantage in several ways. First, communication introduces its own cost in network bandwidth and in time before the trace is completed. Also, such communication can be attacked, giving an attacker a way to defeat the traceback. Perhaps more importantly, if the owner of such machines allows others to use them, then traffic directed to those machines is allowed. This decision then exposes the machines to attacks involving floods of such traffic, potentially preventing use of the traceback facility. For this reason, machines using methods that require

[2] Anyone in a position to help make this happen is invited to contact the author! Here is what we propose as a plausible design for arbitrarily high-speed programmable routers. This design requires a very small amount of special purpose high speed hardware to handle the high speed network interfaces, and an arbitrarily large number of slower general purpose programmable machines, probably running something like FreeBSD or Linux. The special purpose hardware has to divide the incoming high-speed traffic into approximately equal packet streams, which are then distributed to the general-purpose machines. These act as low speed but programmable routers, forwarding packets to the special purpose high-speed hardware connected to the desired output. This hardware then recombines the outgoing traffic from all of the low speed routers into a single high-speed outgoing link. We look forward to the appearance of the proposed special purpose hardware.

additional communication are likely to cooperate only with trackers from inside their own administrative domain. This decision becomes a serious limitation in some important situations, such as traceback over the Internet.

I.2.5 Other Problems

Are there other "special" problems associated with the method?

This is the place for collection of miscellaneous disadvantages relative to other approaches.

I.3 Survey of Traceback Methods

We classify the related work into **link identification**, which we describe first, and **packet filtering**, which we describe second. Within each of these classes we identify a number of subclasses, each of which may be populated by a variety of more specific variants.

I.3.1 Link Identification Methods

These methods were mostly invented for defense against flooding attacks. The presentations of these methods tend to describe how a path can be reconstructed from the victim back to the attacker(s), under the assumption of universal cooperation. A separate report [Cs3 ARDA 1] discusses reconstruction of forwarding paths in the absence of universal cooperation. In the expected situation, all of the cooperating infrastructure is close to the victim. For purposes of flood defense, this is still useful. Unfortunately, this tells the tracker almost nothing about which machines might have generated the packet! The data most useful to the tracker requires cooperation close to the attacker.

The methods in this section each have two parts. One involves collecting data about where the packet was detected along its forwarding path. Cooperating machines along that path do this part. Specifically, the cooperating machines identify links through which a packet was forwarded.[3] This is why we call them *link identification* methods. The second part is what the victim does with that data to reconstruct the forwarding path. We are primarily interested in the data collection part, since we will show (in [Cs3 ARDA 2]) how that data can be used for attribution.

I.3.1.1 Link Testing

Link testing methods are intended to trace ongoing flooding attacks. They identify links through which the traffic of interest flows by testing candidate links, i.e., monitoring them for the traffic to be traced. One simple way to do this is described in [Cisco].

The tracker presumably has administrative control over a router known to be on the forwarding path. That router is reconfigured to assist in determining which incoming link(s) carry the traffic. The tracker, in the case of a flood, might simply ask for a count of packets addressed to the victim arriving on each interface; or for other traffic, provide a description that the router can

[3] By identifying a link we mean identifying both the sending and receiving machines. There are also methods that identify individual forwarding routers, but they can generally be altered to identify links, and this always turns out to be a little better.

use to identify the traffic of interest. Unless the traffic is being generated by the router, it must be arriving at one or more inputs, leading to the next upstream router(s). These routers can then be traced in turn.

Similar techniques can be used to move upstream more than one link at a time. One might, with automated support, simultaneously reconfigure all of the routers on some perimeter around the victim, such as all the border routers of an ISP. One interesting variant of this is described in [Stone]. This paper describes a scheme that reroutes specified packets over tunnels from edge routers to special tracking machines, which can then easily identify the edge routers initially receiving the traffic.[4]

Another variant is described in [Morrow]. This paper describes specific ways to track floods that use randomized source addresses and relies on many of the source addresses being unroutable. It changes the routing tables of routers in the tracker's control first to indicate that the victim is unreachable, which causes packets that reach the affected routers to generate ICMP replies. It further changes the routing tables to indicate that some of the addresses that really are unused (and thus ought to be unroutable) should be forwarded to one particular collection point. This collection point then sees the replies from the reconfigured routers that receive packets with the victim's address as the destination address and unroutable addresses as the source address. Of course, an attack that limits its spoofed source addresses to legitimate addresses cannot be traced by such a method.

It is also possible to do a hostile form of link testing without cooperation from the owners of the forwarding routers, as described in [Burch] (That paper seems to contain the original ideas for most of the methods we describe here!) This method involves attacking (typically flooding) various links, which has the significant disadvantage of attacking other (presumably innocent) traffic. The expectation is that an attack on a link that carries the traffic to be traced will cause a reduction in the rate at which that traffic flows through that link, and therefore a reduction in the rate at which it arrives downstream. This requires cooperation from machines in a position to attack the targeted links. In Burch's paper that cooperation was probably unintentional. It also requires cooperation from machines in a position to detect the loss of traffic, but typically the tracker is already in a position to detect the loss of traffic, i.e., at the destination address. This variant also requires more knowledge of routing than other link identifying methods. The attacking variant is the only one that does not require administrative access to routers. In fact, this was the motivation for its development. Normal link testing could actually be done without cooperation of routers if cooperating machines could be placed at the links to be tested, and if these were fast enough to process the traffic at those links. However we regard this as a different method discussed under the Section on monitors below.

[4] This idea will appear again in the section on deterministic packet marking.

Table I.1. **Summary evaluation of Link Testing (note: + is desirable, – is undesirable)**

Single Packet	– No
Existing Routers	+ Yes
Advance Notice	– Yes Data collection does not start until a traceback problem arises.
Additional Communication	– mostly Yes Normally the operator has to communicate with upstream routers to get results. Methods that use routing tricks avoid this. The attack variant uses lots of additional communication.
Other Problems	– Yes The process is generally manual (controlled by people), hence slow. It may disrupt normal traffic, and generally requires extra work on the part of routers. For these reasons, link-testing techniques are generally considered to be practical only on a temporary (emergency) basis. The need for administrative access to routers will tend to restrict cooperation.

Attacks on Link Testing: Since link testing relies on an ongoing attack, changing the attack pattern with time is likely to confuse matters greatly. An attacker who knew details of the testing (what is being tested and when) could intentionally mislead it.

Since this technique requires administrative access to routers (other than the attack variant), cooperation can be expected to be restricted to the administrative domain owning the routers. To the extent that further cooperation is available it will require additional human intervention, which is slow and inconvenient.

The attack variant does not require access to routers but faces a whole set of problems of its own. It still needs cooperating machines in a position to affect the links to be tested and sensors (generally at the victim, but more generally along the attack path) to detect resulting changes in traffic. It also requires knowledge of network topology and routing. The ability to identify a link further depends on such things as how much bandwidth the cooperating machines have available and details of how routers are configured. For instance the ability to attack might depend on what kind of traffic is used to attack and what kind is being attacked.

I.3.1.2 Itrace

This section and the next (probabilistic packet marking) provide similar data. In both cases, they identify a link over which a sufficient amount of traffic flows. This makes them useful for large aggregates but not single packets.

[Bellovin] describes a method in which routers randomly (with suggested probability 1/20,000) send extra packets to source or destination addresses of the packets they forward to report that the packet traversed a given link connected to the router. With sufficiently many routers implementing Itrace, the recipient of a flood can reconstruct the path(s) of the flooding packets.

Table I.2. **Summary evaluation of Itrace (note: + is desirable, – is undesirable)**

Single Packet	– No Actually, Itrace does identify a single packet but with very low probability, so this is not a good solution for tracking an arbitrary packet.
Existing Routers	– No Although Itrace is described as being done by routers, it seems quite practical to offload the task. If a router could be configured to route packets probabilistically then it could route occasional packets to a separate Itrace machine, which would then forward both the original packet and the trace packet. Alternatively, if a router can be configured to sample packets and send them to an analysis interface, which could be connected to an Itrace machine that would simply send out trace packets.
Advance Notice	+ No
Additional Communication	+ No The trace packets are, of course, additional communication relative to non-Itrace operation, but they are sent as part of normal operation independent of any tracker activity.
Other Problems	+ No

Attacks on Itrace: Attackers can send Itrace packets to make it appear that attacks come from other places, but generally cannot stop the "good" data from telling the victim about the real attacker. The false (attack) trace packets might be distinguished from the real trace packets by authentication techniques, but it is much easier to make up false packets than to check them, so this is another opportunity for the attacker to impose additional cost on the victim.

I.3.1.3 Probabilistic Packet Marking (PPM)

The method described in [Savage] marks a portion of incoming packets. This method is similar to Itrace in that a router has to do some extra work for a small fraction of the packets it forwards, though in this case, the fraction is substantially more than for Itrace (suggested probability on the order of .05). This method replaces (presumably) non-essential data in forwarded packets with data identifying the link over which the packet was forwarded. This allows the recipient to reconstruct the forwarding path(s) of a flood.

Table I.3. **Summary evaluation of PPM (note: + is desirable, – is undesirable)**

Single Packet	– No
Existing Routers	– No As in the case of Itrace, one could imagine offloading this task if the router could probabilistically route packets to a marking machine. However, the increased probability suggests that the marking machine would have to be much faster than an Itrace machine, and for a fast link several such machines might be necessary.
Advance Notice	+ No
Additional Communication	+ No
Other Problems	+ No

Attacks on PPM: Attacks involving spoofed traceback data are described in [Waldvogel] and [Lee-PPM]. In general the two major problems in PPM reliability are:

- The probabilistic nature of the algorithm causes some packets to not be marked by cooperating routers, and these retain whatever marks are given them by the senders. Attackers can simply mark their original packets to intentionally mislead the traceback mechanism.

- In an effort to avoid expanding packets, PPM uses (supposedly) underutilized space in current IP packets. But this is not enough space for all of the desired link identification data, so the marks in a single packet are used to provide only partial data. However this also leads to errors, particularly in the presence of large numbers of paths.

[Song] describes attempts to fix some of the problems. Our view is that even if all of the problems with PPM are fixed, probabilistic marking has no advantage over Deterministic Packet Marking (DPM), which we will discuss later, whereas deterministic marking has significant advantages over probabilistic marking, particularly for a single packet. High-speed commercial routers do not currently support *any* of the marking methods. None of them appears to be significantly more difficult to support than any other. In the unlikely event that we can influence router manufacturers to support new features it would be best to ask for DPM, which is more useful.

The supposed advantage of probabilistic marking over deterministic marking is that it does not require additional space in the packet. This is considered to be a problem not because of the loss of payload bandwidth but because the space will increase with each hop and this is thought to be expensive for routers. That problem can be solved by allocating the space all at once at the first marking router. The marking itself seems trivial, comparable in complexity with decreasing TTL. In fact, the work to allocate the extra space in the packet at the first router is also relatively small. A more realistic objection is that the extra space is likely to cause the packet to exceed the MTU on the next link. Even this *should* not be a problem since TCP/IP was designed to deal with that case. In the current Internet it actually is something of a problem because of widespread misconfiguration. See [rfc2923] and [MSS] for further details. Note the same problems occur

with other protocols that need to steal a little bandwidth, e.g., tunneling. There are commonly used "work-arounds" (perhaps more accurately described as "I have to break my network in order to work with your broken network!") for such problems.

I.3.1.4 Deterministic Packet Marking (DPM)

DPM is not well represented in the literature. Two methods, described below, appeared in 2003, but these do not fairly represent our idea of what DPM is. They both use the PPM trick described above to avoid expanding packets, and in the process they lose much of the benefit, as we explain below. Other than the references we cite below, the only references we find in the literature are PPM papers that mention the possibility of DPM and claim that it is impractical to expand packets. We have explained above why we disagree with that assessment.

We therefore describe DPM in more detail than the previous methods. We believe that, other than the IP record route option, also described below, we are the only ones to have ever actually implemented a DPM method. We may be the only ones to have implemented *any* packet marking method.

In DPM, routers mark all forwarded packets with link identifying data. With PPM, multiple routers on the path overwrite the same data, and each packet identifies at most one link. With DPM, each cooperating router *adds* link identifying data to the packet, and each packet ends up with data that identifies *all* of the links (under universal cooperation) that it traversed. Packets traversing the same path are marked the same. The simplest and oldest example of a similar capability is the IP record route option. IP routers are supposed to add their own addresses to such an option as they forward packets. The Record Route option actually records a router rather than a link, but if the option had room for two addresses, it would record the first two routers, which would identify a link.

The first problem with the IP option as a traceback method is that it is the choice of the sender to include the option. The IP option allows traceback of senders who wish to be identified, but that is not very useful. Cooperating routers inserting the option into packets that do not already have it could solve this problem.

The next problem is that routers are supposed to leave the record route option unchanged if the space allocated for it is already filled with data. An attacker could therefore spoof any data he wanted to. The solution to this problem is that cooperating routers should be configured to know which of their neighbors are trustworthy and accept trace data only from such neighbors. Data from other neighbors should be overwritten. This strategy gives us the address of the last router on the path that was preceded by an untrusted neighbor.

In the case where the first cooperating router is not connected to the last in a continuous path of cooperating routers, this loses some data, namely the links traversed before the packet reached the last untrusted router. Unfortunately, as is shown in [Cs3 ARDA 2], this is actually the most valuable data! The best solution to this problem that we have found so far is tunnels between non-adjacent cooperating routers, an idea that appears to have not been explored before in this context.

Similar techniques were mentioned under link testing. In effect, a cooperating router must determine which cooperating router will be the next to receive a given packet, and encapsulate that packet in a tunnel to that next router. In the most general case, this leads to an overlay network of cooperating routers. The design of such a network reflects trade offs between the efficiency of routing and what traceback data is available at what places.

It should be mentioned that cooperation in a given method is not always a binary property of a machine. In this case, an ISP might be willing to route packets addressed to your network via a tunnel to another router cooperating with you, but still not be willing to route other packets that way.

Another problem is that the trust relationship described above is transitive. Accepting a mark from a neighbor means not only trusting that neighbor but also trusting those trusted by that neighbor. A single compromised router mistakenly trusted by its neighbors can generate any traceback data and that data will be passed along and accepted until it reaches a link where one neighbor does not trust the other. For this reason it is better to gather at least some additional downstream data for purposes of validation. We show below how that data can be used.

Cohen-Responsibility describes a form of packet marking that gathers complete path data in a very compressed form, encoding each link in just the number of bits to specify one of the neighbors of the machine that received the packet on that link.[5]

As will be seen in [Cs3 ARDA 2], the most valuable data for traceback is the IP address at the sending side of the first identified link. The compressed path does not make extraction of that data convenient. In small networks with static routing it is sufficient to simply store the association between paths and their origins. In networks with single path routing, such information can be easily generated by sending a ping to each address and storing the path recorded on the reply packet.

[CS3-Forgery] suggests a modification that is more suitable in the general case. The sending IP address of the first recorded link is added to the path data. The compressed path data is sufficient for reconstructing the initial address (in fact, the entire path) but this reconstruction requires either information about network topology or additional communication to provide such information. An easy test to see that the initial IP address does actually correspond to the recorded path is to send a ping packet to the IP address and see whether the reply comes back with the same path. This test assumes a single path and a stable route.

Note that a router can still forge paths that pass through itself. [Song] describes some techniques that are actually meant to address problems in nondeterministic packet marking, but seem applicable to this problem as well.

[Yaar] describes PI, a deterministic marking method that is in some ways a compromise between deterministic and non-deterministic marking. It is deterministic, but like the non-deterministic methods, avoids expanding the packet and generally provides less complete traceback data**:**

[5] This is the earliest reference we have seen for this scheme. It was reinvented and implemented as part of a packet flooding defense described in [CS3-DDoS]. We now see some ways to do a little better, but as a practical matter, this is probably not necessary.

- Like PPM, there may be a residue left over from the initial marks. Unlike PPM, the residue is a particular set of bits that survive a given path. Therefore, different marks may in fact come from the same origin.
- The same marks may appear on packets that took different paths.

In general, then, the marks that arrive with PI do not identify a single link as the other methods do. They do still identify a set of possible links, thereby complicating the analysis below.

The objective of PI is not actually to identify links, but to allow the recipient of a set of packets to group them by forwarding path. This turns out to be useful in defense against floods. If routing were known to be stable and deterministic (single path), the recipient could test the consistency of a possible origin with the marking supplied by PI by using the ping trick described above.

[Belenky] describes another deterministic marking method. This method is actually meant to determine packet origins. Belenky correctly points out that the first link is the one with the important data and proposes to keep only that data. He seems not overly concerned with the fact that one compromised or misconfigured router can then generate any mark. Like PI and all of the PPM methods reviewed above, Belenky stores data in existing fields of IP packets. Since the available space is insufficient to identify even one link, Belenky's method requires multiple packets in order to identify the first link. Reconstructing the data from multiple packets presents the same sorts of problems described under PPM. This method is not completely deterministic, since it nondeterministically chooses which part of the link data to store in a given packet.

Some commercial routers can be configured to do a very limited form of deterministic path marking. Cisco IOS allows routers to alter the TOS field, which could be used by a small transitively connected set of routers, which we call a *neighborhood* of routers to identify up to 64 places where a packet enters the neighborhood. Possibly other commercial routers can be programmed in similar ways. This method could be used in conjunction with routers that do real path marking to allow small clusters of legacy routers to cooperate in a larger neighborhood. That is, the old routers would use TOS marking at entry into the neighborhood, and at exit, a more capable router would incorporate that mark into the real path data.

As in other methods above, it is also possible to get away without changing routers in the case where a small portion of the traffic coming into a router can be identified as worthy of traceback. In this case, the router could simply route that traffic to a separate lower speed machine that does the marking. This machine could also do the tunneling mentioned above.

Table I.4. **Summary evaluation of DPM (note: + is desirable, – is undesirable)**

Single Packet	+ Yes
Existing Routers	– No A minor exception is TOS marking discussed above.
Advance Notice	+ No
Additional Communication	+ No additional communication can be useful for validation
Other Problems	+ No

Some people consider it a significant problem to increase the packet size. We regard this as minor. This is discussed above under probabilistic marking. The tunnels used for communicating between non-adjacent cooperating machines may be too expensive to allow large amounts of traffic, and this may serve as a reason to restrict cooperation.

Attacks on DPM: An attacker who controls a trusted router can forge any path up to that router unless some further authentication scheme is used. Those paths that disagree with the routing algorithm can be recognized by a simple check as described above. A router that trusts data from an attacker effectively allows that attacker to act like a compromised router. Authentication methods could be used, but these add significant cost in the form of processing time and space in the marked packets.

I.3.1.5 Tunneling from Cooperating Machines to Tracing Machine

The overlay network described in section I.3.1.4 suggests an additional method. Instead of tunneling to the next cooperating router, suppose we tunnel to the last one. This is particularly easy in certain common cases, such as the one where we are only interested in tracing packets sent to our own network. We simply ask the cooperating routers to encapsulate traffic addressed to our network in a tunnel directly to a router in our network. Note that the remote cooperating routers do not have to do any marking, which means that they can be currently available commercial routers. The receiving router can tell from the tunnel on which a packet arrives which cooperating machine forwarded it. In fact, the remote cooperating router can identify the link on which it received a packet by using different tunnels to carry packets that arrive on different links.

The tunneled traffic might carry authentication data but still be unencrypted so as to allow other machines to trace the encapsulated packets. However, at the end of the tunnel, the original packets emerge without the tunnel encapsulation, which is now carrying the traceback data. If traceback is to be done after that, then something must be done beforehand to save the data. One possibility is marking in the normal sense. Another is to summarize the traceback data and store the result for later use.[6]

[6] In either case, the receiving machine will probably have to be more programmable than a commercial router. Even if the end of the tunnel is the host to which the original packet was addressed, some additional processing must be done in the operating system to make use of the attribution data, since it will not be visible at the application level. How the attribution data is stored and accessed after it arrives, however, is outside the scope of this paper.

If no forwarding paths pass through more than two cooperating machines then this method is the same as the overlay network. We should therefore consider the difference in the case of a packet that passes through more than two cooperating machines. In the overlay network, the first machine sends the packet in a tunnel to the second machine, which then adds a mark and sends it through another tunnel. Here the first machine sends the packet in a tunnel directly to the final cooperating machine. An intermediate cooperating machine receiving a packet already encapsulated could try to verify that the encapsulation is not forged, and if satisfied, simply forward the packet unchanged. A more straightforward solution is to re-encapsulate the packet. The receiving machine would then have additional work to do, but would find that the packet had traversed both machines. In effect, we use encapsulation as a form of marking. It is, however, an inefficient method, especially if performed at many intermediate points. This method therefore seems best suited to the case of sparse cooperation, which we expect to be a common case.[7]

Table I.5. **Summary evaluation of Tunnels (note: + is desirable, – is undesirable)**

Single Packet	+ Yes
Existing Routers	+ Yes This method assumes that the router supports tunneling and that packets to be traced arrive at a low enough rate to be tunneled.
Advance Notice	+ No Except that the tracker might want to change the set of packets to be traced.
Additional Communication	+ No
Other Problems	+ No This method may be too expensive to allow large amounts of traffic, and that may serve as a reason to restrict cooperation.

Attacks on Tunnels: none

I.3.1.6 Source Path Identification Engine (SPIE)

SPIE and Remote Monitors (described in section I.3.1.7) are what might be termed "logging" methods.

SPIE, described in [Snoeren], requires routers to remember, possibly for a very limited time, data that allows them to answer with high confidence the question "*Did you recently forward the packet P?*". The forwarding path of a single packet can be reconstructed by querying such routers soon after the packet is observed. More recent work (private communication) moves the processing from the router to a specialized machine observing traffic on a link. This method can be viewed as a special case of Remote Monitors, described in section I.3.1.7.

[7] This use of tunnels is similar to that proposed in [Chang]. They seem to have had in mind a somewhat different use from ours. They propose to build tunnels after an attack is recognized, which would only make sense for ongoing attacks. Also, they seem to imagine a situation where it is possible to build a tunnel from anywhere.

Table I.6. **Summary evaluation of SPIE (note: + is desirable, – is undesirable)**

Single Packet	+ Yes
Existing Routers	– No As noted above, this task could be performed without the router. In order to support very high speeds it would be necessary to use special purpose hardware.
Advance Notice	+ No
Additional Communication	– Yes As in other monitoring methods, one has to query the monitor.
Other Problems	– Yes The storage required is proportional to the number of packets one wants to record. This method achieves high compression, on the order of one or two bytes per packet, but this is still a lot of storage per unit time for a fast router.

Attacks on SPIE: Attackers can attack the query/response communication, either the traffic or the endpoints. For that reason access to traceback data will normally be restricted to the administrative domain owning the routers, and possibly a few other trusted places.

I.3.1.7 Remote Monitors

General purpose machines controlled by the tracker can be distributed throughout the network to listen to traffic passing by. They record whatever they are programmed to, and then answer questions about the recorded data. What they are programmed to record depends on what the tracker expects to need, but the rate of data to be processed, the amount to be recorded and the time for which it is retained are limited by the speed and memory of the machine.

Table I.7. **Summary evaluation of Remote Monitors (note: + is desirable, – is undesirable)**

Single Packet	+ Yes Of course, one can only expect to trace the packets that the sensors have been instructed to look for, which is, generally, a small subset of all packets.
Existing Routers	+ Yes
Advance Notice	– Yes
Additional Communication	– Yes One has to query the monitor.
Other Problems	+ No

Attacks on Remote Monitors: The same considerations described under SPIE apply here as well.

I.3.2 Packet Filtering Methods

The methods described in this section filter packets with false source addresses at routers. The relationship between these methods and the Level 1 attribution problem is analyzed in more detail in [Cs3 ARDA 2], which describes the design of our new Level attribution method.

I.3.2.1 Ingress Filtering

[rfc2827] suggests that routers filter packets with obviously false source addresses. “Obvious”, in this case, means that a given interface is only supposed to carry traffic to/from an a-priori known fixed IP range. This method is a special case of “route based” filtering, discussed next.[8]

Table I.8. **Summary evaluation of Ingress Filtering (note: + is desirable, – is undesirable)**

Single Packet	+ Yes
Existing Routers	+ Yes
Advance Notice	+ No
Additional Communication	+ No
Other Problems	– Yes This method limits the possible origins of a packet to either the set of machines with addresses in the range accepted by the filter or any machine that is not restricted by such a filter. When many machines are not filtered, this is a very large set of possible origins. Another problem is that it may not even be easy to find out where such filtering is deployed.

Attacks on Ingress Filtering: none

I.3.2.2 Route Based Filtering

[Lee-DPF] describes a generalization of ingress filtering in which routers filter packets with source addresses that are inconsistent with routing data. The generalization involves replacing the word “obvious” above with much more sophisticated reasoning based on global routing data. We note that it is not clear yet how to obtain sufficient routing data for the Internet. [Li] describes a protocol by which such data could be collected, but we again have the usual problem of how well partial compliance (in this case, using the protocol to generate the needed data) solves the problem.

To the extent that route based filtering is practical, it could be combined with other link identifying methods to allow a router to filter on the basis of not only the fact that this packet is arriving at one of its own interfaces, but to simulate a filtering router further upstream.

[Lee-DPF] presents many results on how effective filtering would be if it were done by, e.g., a randomly chosen half of all routers in the Internet, or a set of routers chosen by a greedy algorithm to cover all links. For example, it reports that cooperation from the “right” 20% of AS’s (that is, we get to choose the 20%) localizes every IP address to 5 AS’s. Unfortunately, we do not expect in practice to have much control over which places cooperate. All of the analysis is based on simulation at the level of AS’s. Of course AS’s can be large, so determining the AS of origin might not be very satisfactory. It is not clear whether or how finer grained routing data can be obtained outside one's own AS.

[8] This idea seems first to have appeared in 1996 in [Cohen-Forgery].

***Table I.9*. Summary evaluation Route Based Filtering (note: + is desirable, – is undesirable)**

Single Packet	+ Yes
Existing Routers	– No In some cases the set of filters will be simple enough to implement with normal filtering methods.
Advance Notice	+ No
Additional Communication	+ No
Other Problems	– Yes The main problem is obtaining routing data. The next problem will be maintaining the data in real time under routing updates. There may also be as yet unknown implementation difficulties in performing the filtering. And, like ingress filtering, it may be difficult to determine what filters are actually in place.

Attacks on Route Based Filtering: Since it is not yet clear what problems will arise in trying to implement this method, it is also not yet clear what an attacker might do to magnify those problems. One possibility is causing routing updates and preventing the filtering routers from learning how these affect what should be filtered. The routers might then either filter legal packets or, in an effort to avoid that, forward many illegal packets.

I.3.3 Summary of Analysis

We now summarize the analysis above in a single table on the following page. Note that in addition to yes or no answers, + is used to indicate that this is the preferred answer, – that it is not. Techniques that we consider to be closely related are grouped into sections.

Table I.10: **SUMMARY OF METHODS AND CHARACTERISTICS**

Method (preferred answer)	Single Packet Trace (yes)	Works with Existing Commercial Routers (yes)	Advance Notice Required (no)	Additional Communication Required (no)	Has Other Problems (no)
Link Identification methods					
Link Testing	no –	yes +	yes –	varies	yes –
Itrace	no –	no – [1]	no +	no +	no +
PPM	no –	no – [1]	no +	no +	no +
DPM	yes +	no – [1]	no +	no +	no +[2]
Tunneling	yes +	yes +[3]	no +	no +	no +
SPIE	yes +	no – [4]	no +	yes –	yes – [5]
Monitors	yes +	yes +	yes -	yes –	no +
Packet Filtering methods					
Ingress Filtering	yes +	yes +	no +	no +	yes – [6]
Route Based Filtering	yes +	no – [7]	no +	no +	yes – [8]

Notes:

[1] This may be achievable by using (possibly nonstandard) router features to route a subset of packets to another cooperating machine.

[2] Marking across non-cooperative infrastructure requires use of tunnels.

[3] It is not clear whether tunneling will work with high-speed routers.

[4] Current work attempts to monitor links with a separate machine. Very high-speed links would still require new special purpose hardware.

[5] SPIE has to deal with the problem of trading off between additional memory vs. time.

[6] Ingress filtering has a very poor effectiveness to degree of cooperation ratio.

[7] Current routers are able to filter, and in some cases there will be few enough filtering rules to use this mechanism without unacceptable cost. However, we expect that will not always be the case. In the general case, filtering is similar in complexity to routing.

[8] Route based filtering assumes that relevant routing data can be obtained.

Chapter II

Techniques for Level 2 Attack Attribution

II.1 Problem Statement

Our model is that there is a causal relationship among activities in computers, i.e., that one activity in one computer can "cause" another activity in that computer or, through communication, in another computer. Given as input some activity and the computer in which it occurs, the goal of "Level 2 attribution" is to find the beginning of the "causal chain" that leads to that activity. The result is again an activity along with the computer in which it occurs. The term "causal chain" suggests a linear sequence of machines, but in fact the controlling relationship among activities can be many-to-many - not only can one machine control many, but one machine can be controlled by many. Therefore, there may be many controlling paths between two machines. The beginning of the causal chain might be the input activity itself (meaning that no other activity causes that activity), another activity in the same computer or another computer or multiple activities in multiple other computers.

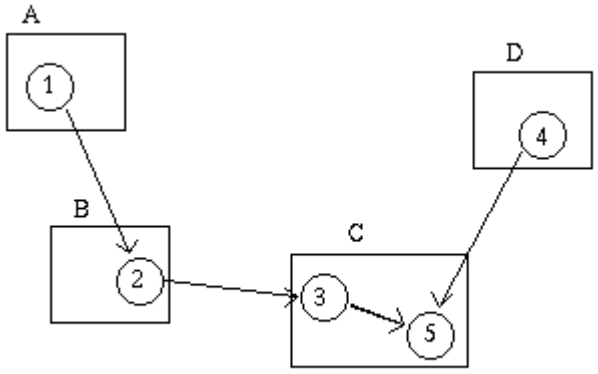

Figure II.1. Causal Relationships Between Computer Activities

Figure II.1 depicts computers as boxes, activities as circles inside those boxes, and the causal relation as arrows from the causing activities to the activities that they cause. In Figure II.1, an attacker at A breaks into B (activity 1), then uses B to break into C (activity 2) and run a DDoS slave there. The DDoS slave is activity 3. Then the attacker uses D to send a trigger to C (activity 4), and this in combination with the DDoS slave (activity 3) causes an attack (activity 5). Notice that these activities do not all happen at the same time. In particular, activities 1 through 4 have all stopped by the time activity 5 starts. The tracker initially sees the attack (activity 5). The tracker's objective is to determine that this activity is ultimately caused by activities 1 and 4 in computers A and D.

The most obvious approach to finding the beginning of a causal chain is to trace back one step at a time from the activity to be attributed. Most of what we describe here is concerned with how to take that next step backward in the causal chain. The terms **traceback** and **trace** refer to the activity of trying to take the next step back. Some techniques are capable of taking more than one

step at a time. There are no techniques that necessarily lead directly to the primary controlling host for reasons described below (section I.1.3). Furthermore, as we will show (section II.5), taking one step back along the causal chain is not actually guaranteed to get "closer" to the goal.

The rest of this section is devoted to further clarifications.

II.1.1 What does "Attacking" have to do with the Problem?

The BAA is about "attack attribution", but the problem statement above does not mention attacking as opposed to other behavior. Our view is that we have defined "Level 2 attribution". If the behavior to be attributed is viewed as an attack (which we view as a subjective matter), then attribution of that activity can be considered to be attack attribution. Attackers are, perhaps, more likely than others to try to make attribution difficult, but non-attackers may also wish to remain anonymous. Just for convenience we call the party trying to do attribution the **tracker**. The party he is trying to identify and also any party trying to prevent attribution we call the **attacker**. These are clearly adversaries, but either might be in the right or wrong. We devote considerable attention below to issues of what measures and countermeasures these adversaries can take against each other, but little to the distinction between good and bad behavior.

II.1.2 What are "Hosts" and what do they have to do with the Problem

The use of the term "host" in the BAA suggests that not all machines are hosts and those that are not somehow do not matter. We believe that the intent of the BAA is to not consider machines such as routers to be involved in an attack just because they forward the packets that are involved. A complete causal chain as described above would probably be dominated by such forwarding behavior, which is the focus of Level 1 and not Level 2.[9]

A router could be involved in behavior relevant to Level 2, for instance, if it were used as a reflector to send ICMP packets to a victim in response to packets sent by an attacker with the victim's address as the source address. A tracker would be even more interested if a router were controlled by an attacker, because the attacker then has access to a large class of additional controlling behaviors. We therefore conclude that the problem really has nothing to do with "hosts". The primary controlling host could really be any machine that can send network traffic. We use the terms **machine** and **computer** interchangeably for such machines.

II.1.3 What is the "Primary" Controlling Host?

The word "primary" in the BAA motivates the word "beginning" (of the causal chain) in the problem statement above. Ideally the "primary controlling host" should be the one connected to the keyboard touched by the human to eventually be identified by Level 3 attribution. Short of seeing the controlling fingers on the keyboard, however, it is nearly impossible to be sure that a given activity is not caused by another activity at another computer. The control channel might not be observable by the tracker. The best result one can expect is to determine with reasonable certainty that an activity observed on one computer is caused by an identified activity on another

[9] Level 1 is analyzed in Chapter I of this report, in [Cs3 ARDA 1] and [Cs3 ARDA 2]. A full analysis of Level 1 techniques also appears in [Level1]

identified computer, or that the observed activity is not caused by any other computer by means that the tracker can detect.

II.2 Division of the Problem into Cases

There are several ways to control a computer. Figure II.2 shows the different modes by which a computer can be controlled by an attacker. Each mode is indicated by a particular kind of access that the attacker is able to acquire and exploit. Moreover, the modes of control are ranked by the degree of control that the attacker has over the victim's computer. As the degree of control available to the attacker increases (as one moves to the right in Figure II.2.), the harder the task of the tracker becomes.

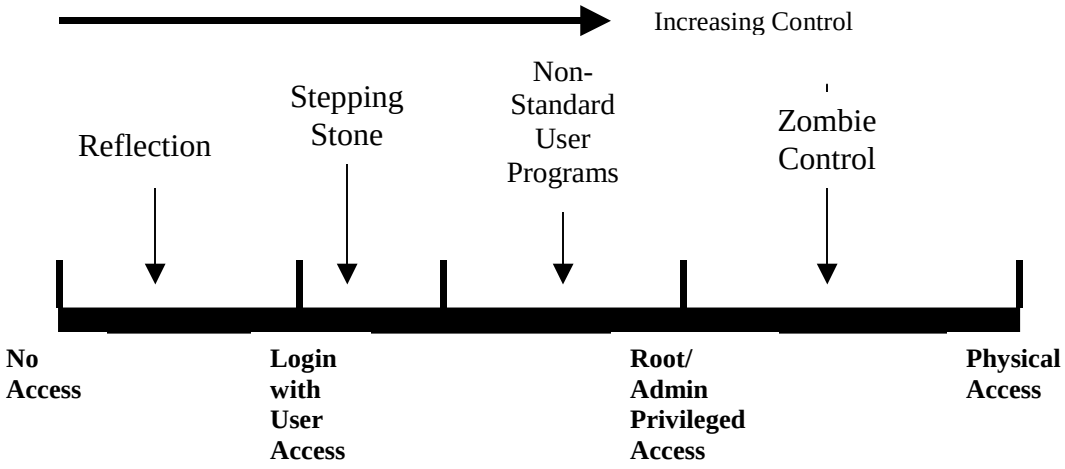

Figure II.2. **Modes for Attackers to Control Machines**

To elaborate on Figure II.2., the modes in which a computer can be controlled may be described as below:

a) **Reflection Control:** At this level, the attacker has no login access, but is able to communicate through a network with the computer that is being controlled. Further, the attack is able to exploit normal programs and services that are standard to mount attacks.

b) **Stepping Stone Control:** There are standard programs (such as *telnet*, *rlogin*, and *ssh*) that can be used by attackers to control computers. Note that Stepping Stones require login access to the machine being controlled. By definition, stepping stones must use standard, benign, programs and must maintain active, real-time connections to the computer being controlled.

c) **Non-Standard User Software Control:** Once an attacker has login access, it is possible to install non-standard programs that might control other computers in non-standard ways. Programs installed under this kind of control are limited by the fact that they must run under ordinary user privileges.

d) **Zombie Control**: Most computer installations distinguish between normal user capabilities and "root" or "admin" privileges. Administrators typically have vastly more rights and capabilities in a system for good reason. An attacker who is able to compromise a machine to get administrator privileges essentially controls the "mind" of the computer by being able to install arbitrary programs and services. Since not all operating systems have the term "root" or "admin" as a notion of privilege, we define such control as Zombie Control.

e) **Physical Control:** Finally, a computer can fall under the physical control of the attacker. This is the ultimate level of control that we examine in this document.

The tracker's difficulty in identifying how one activity controls another depends on which of the types of control discussed in Figure II.2. is involved. An attacker would naturally prefer to use control mechanisms that cause difficulty for the tracker. However, the attacker's choice of control mechanisms is limited by the amount of control that he or she has over the machines used, i.e., the degree to which these machines cooperate with the attacker.

It would make sense to also consider the degree of control the attacker has over the controlling computer, but categorizing by this degree of control would not help the tracker, since the tracker has no data about the controlling computer (including whether such a computer even exists) until the tracker solves the problem at the controlled computer. Once the tracker determines that there is a controlling computer, the nature of the control may lead to the conclusion that the attacker has a certain degree of control over that controlling computer. Note that an attacker with a certain amount of control over a computer might use a technique that requires less control just to confuse the tracker.

We now analyze the different categories of control in much more detail.

II.2.1 Reflection Control

A computer communicating with IP (Internet Protocol) must run some programs that accept input from the network. Typically anyone on the network, including any attacker, can send such input, which will cause those programs to react in well-defined ways. It would be best if these reactions could not be used for malicious purposes, but in the current Internet, using IP, this is not the case. The mechanisms that are needed for communication can also be used to attack. In a **reflection attack** the attacker sends packets to an innocent machine, which we call the **reflector**. The reflector, acting according to specification, sends other packets, which we call the **reflection packets**, to the victim. We define a **reflection attack** as an attack in which all of the following conditions are met:

- the attacker sends packets to the reflector with the victim's address in the source field (which is already an illegitimate behavior)
- the reflector sends packets to the victim in reply to the attacker's spoofed packets
- the attacker uses no knowledge about the reflector other than the fact that it acts according to IP specifications. Specifically, passwords are not needed and vulnerabilities are not exploited. The attacker is not considered to have "broken into" the reflector or the victim.

We believe that the definitions above agree well with previous usage in the literature, e.g., [Paxson]

It is reasonable to attribute such attacking behavior not to the reflector, but to the agent responsible for sending the packet with the falsified source address. More precisely, Level 1 attribution would find that the reflection packets were generated by the reflector, and it is the job of Level 2 to attribute that behavior to the machine that generated the packets with falsified source addresses.

There are some cases that do not satisfy the definition above but are similar from the perspective of Level 2 attribution. One of these is an attack that makes use of a vulnerability in some TCP implementations that allows the attacker to spoof the victim for an entire connection. The attacker can then cause a server to send a large amount of data to the victim. The cost to the attacker is the submission of a much smaller amount of data to the server, in the form of TCP acknowledgments. This is not a reflection attack because the attack relies on a vulnerability and the knowledge that the server has that vulnerability. Nevertheless, the attribution methods that apply to reflections seem to apply to this attack as well.

Network Address Translation (NAT) is a similar case. NAT does not use falsified source addresses, but it is susceptible to similar attribution methods. What all of these cases have in common is a very simple relation between incoming packets and outgoing packets. NAT normally occurs in places where different machines can no longer be identified by IP address from the outside. Normally, at these places there are only a few machines sharing an IP address, typically in the same location and belonging to the same entity. Identifying the NAT machine is almost as good as identifying one of its clients. (There are techniques that can be used to identify one of the clients if that should prove useful, e.g., related to OS fingerprinting.)

II.2.2 Stepping Stone Control

Many computers accept remote login, which is meant to allow legitimate remote users to run additional programs that are installed on those computers. Unlike reflections, use of remote login requires some knowledge about the machine that attackers are not supposed to have, i.e., a user name and password. In some cases the attacker might have legitimate access to the remote computer. If not, we consider the attacker to have "broken into" that computer.

A person who logs into a computer is supposed to have more control over that computer than a person who does not log in.[10]

A user who logs into a remote computer can use that computer to attack a different computer in a number of ways. The standard programs that the attacker uses from a computer to log into a remote computer can also be used on the remote computer to log into yet another computer. Again, it is the job of Level 2 to recognize that the behavior originally identified as emanating from one computer is really controlled by remote login from another computer. The term "stepping stone" was originally used in the literature to refer to a specific small set of command

[10] It is certainly possible to install programs that allow such additional control to anyone on the network, but such programs are not standard, for obvious reasons. They are sometimes installed by attackers who gain illegitimate control over machines.

line interfaces, including *telnet*, *ssh* and *rlogin*. When limited to this set, it is fair to say that the traceback problem has been attacked with reasonable success. In this category we will also discuss other standardized widespread mechanisms meant to allow a user on one machine to control another machine in real time for legitimate purposes. These mechanisms include anonymization software, which is meant to prevent traceback. Of course, what software is "standardized and widespread" varies considerably with time, and anonymization software is well on the way towards satisfying that description.

II.2.3 Non-Standard User Software Control

Standard programs intended for legitimate purposes can be used to generate or retrieve other "non-standard" programs. The term "non-standard software" is meant to convey the idea that the controlled machine is doing something other than running well-known, widely recognized software that often conform to specific protocols. Note that any non-standard programs installed are still restricted by the operating system software. The possible restrictions vary with operating system, and the actual restrictions vary within that range from one computer to another due to differences in configuration. By definition, in this category of control, the programs installed have normal user privileges, and we, therefore, assume that the installed software cannot do such things as send arbitrary packets and read and write arbitrary files.

Typically, someone with enough access to a computer to use it as a stepping stone has enough access to install and use non-standard software. Possible reasons why the attacker might not be able to generate or retrieve programs include lack of file transfer service and compilers, or insufficient storage allocation. The attacker may lack the knowledge to retrieve and run non-standard programs, or, if these programs do not serve the purpose, the attacker may lack the ability to create programs that do. The attacker may also lack motivation, i.e., malicious intent.

One important difference between stepping stone control and non-standard, user software control is that an observer from outside the machine can readily understand stepping stone behavior since it follows well known rules, whereas non-standard software behavior does not follow well known rules and therefore cannot be reliably understood without observing the program that generates that behavior. This difference makes Level 2 attribution significantly more complicated for non-standard software. For example, the tracker may see a machine communicating with several others in what appear to be normal ways, e.g., web browsing or mail. Any of these communication channels might actually be used to control this machine or used by this machine to control the other machines with which it communicates. Even if the communication appears to be unencrypted, it could be using covert channels, such as the times at which packets are sent or steganography.

Another important difference is that stepping stone behavior, as part of its "real time" requirement, exhibits a high temporal correlation among the connections along the causal chain. That is, the controlling activity occurs more or less at the same time as the controlled activity. Non-standard software activity may or may not exhibit such temporal correlation. The controlling communication can cause behavior at arbitrary later times, i.e., a behavior observed now could be caused by any past communication. Therefore, access to historical data is often required to do Level 2 attribution for non-standard software behavior. It also means that some controlling machines might not even exist at attribution time.

II.2.4 Zombie Control

If the attacker manages to have privileged access to the computer being controlled, or through some other means is able to circumvent all the constraints imposed by the operating system, the next stage of control is reached, which we call “zombie control”. Zombie control is as much control as one can have over a machine without access to the physical hardware. The term “zombie” is often used for a machine used to mount (particularly denial of service) attacks on command. In this context, we use the term to denote that the attacker has ultimate control over the computer at the highest level of privilege available.

The ability to send arbitrary packets enables such behaviors as reflection attacks. The ability to listen to all packets arriving at the machine is perhaps more interesting still. Many machines are in a position to observe packets that are addressed to other machines. The result is a new set of covert communication channels. Machine M1 can communicate with machine M2 by sending packets to machine M3, if the attacker knows that packets sent from M1 to M3 will pass by M2 and if M2 can be caused to observe them. From the tracker's point of view, the consequence is that any packet could carry information not only from the sender to the destination, but to any point along the path. As an example, M1 might now send ping packets to M3 in order to communicate with M2. As before, the data could be hidden in the packets or in the timing.

II.2.5 Physical Control

Physical control is the ability to unplug the machine, unscrew the cover, remove or add hardware, etc. Physical control is more powerful than zombie control. For instance, a person with physical control can always remove the zombie control of another person, even if the one with zombie control obtained that control earlier. We believe that physical control is actually less useful to the attacker than zombie level control. One reason is that it is much more difficult to obtain, and therefore much more rare. It also implies physical presence. Physical presence of the attacker is a big advantage for the tracker. By locating the machine the tracker can locate the attacker. The machine at the beginning of the causal chain actually is under the physical control of the attacker. The motivation for Level 2 attribution is actually to find the attacker by finding the machine(s) over which the attacker has physical control. Similarly, the reason for using zombies, stepping stones, etc. is to prevent the tracker from finding the machines over which the attacker exerts physical control. If the attacker were to use some technique that requires physical control, the tracker would know that the attacker was (or at least had been) at the location of the controlled machine.

II.2.6 Likelihood of Various Levels of Control

A behavior of a given machine observed by the tracker implies that the attacker has *at least* enough control over that machine to cause the observed behavior. An attacker with more control than implied by the behavior could use that behavior to mislead the tracker. For example, an attacker with zombie control over machines M1, M2 and M3 could make it appear that the *ssh* session emanating from M1 is controlled by an *ssh* session from M2 even though it is really controlled from M3 through a hidden channel. How realistic this threat is depends on how likely it is that the attacker has different amounts of control. We therefore briefly consider that issue in this subsection.

As mentioned above, all that is required to move from stepping stone level control to non-standard software level control is access to file transfer software, adequate storage space, and the knowledge to obtain and run the software desired. Most machines connected to networks offer the first two to any user who logs in. At one time (ancient history in Internet time) the most common sort of control available to both legitimate users and attackers was the ability to log into a machine on the network and run the programs installed there.[11]

Most of the machines in the current Internet serve a single user. Legitimate control is normally physical and most illegitimate control is zombie level control over machines that were highly insecure from the time they were connected to the network. Therefore it seems very likely that an attacker using a machine as a stepping stone actually has zombie level control. Our view is, therefore, that the important problem is now zombie level control. The other levels are instructive, however, because they allow us to relate different attacker capabilities to the problems they pose for the tracker and what he can do about them.

II.3 Classification of Methods

Level 2 attribution is a process that draws conclusions from data available to the tracker. What data is available depends on the degree to which machines in various places cooperate with the tracker, or equivalently, the degree to which machines are controlled by the tracker. It is possible for the owner of a machine to grant to a tracker (or attacker) various levels of control as identified in section II.2. For instance, the tracker could be given the ability to read certain data that would normally require zombie level control without the ability to read or write any data, as would be implied by real zombie level control.

For purposes of Level 2 attribution, the data of interest is always related to the behavior of machines in the causal chain. Different methods make use of different data. We find it useful to classify methods by the data they use as below:

a) *Internal Monitors***:** Programs running inside the machine..

b) *Logs***:** Historical data available at the machine.

c) *Snapshots***:** A complete copy of the state of the machine as of a particular time.

d) *Network Monitoring***:** Data related to communication between machines.

e) *Reaction to Tracker Activity***:** Data related to attacker reaction to tracker behavior.

In this section, we describe the details about each of the classes.

II.3.1 Internal Monitor

Internal Monitoring refers to observation of the activity of a machine from “inside” that machine, using monitoring programs intended for that purpose. Some of these programs require zombie

[11] At that time there were many more users than machines, so typical machines served many users. Many users per machine requires good protection of users from each other, which in turn requires that zombie level control is difficult to obtain. Along with more users per machine, there was much more administrative expertise per machine than is now the case. The most common means of illegitimate control was insecure passwords.

level control and others only non-standard software level control. Within a private network the tracker can expect cooperation if the tracker works for the network owner. Typically, the victim of an attack is willing to cooperate. In fact, the tracker is normally either working for the victim, for the victim's ISP (which may also be a victim) or for law enforcement. For purposes of tracking in the Internet, the tracker can expect little cooperation from most machines, and that includes most of those in the causal chain. Machines close to the victim are more likely to cooperate than others, mainly because they are also likely to belong to the victim. The machines close to the victim tend to be attractive targets for the attacker, since more damage can be done to the victim by controlling machines close to the victim.

II.3.2 Logs

A Log is data recorded by a machine specifically for the purpose of recovering information about past activity. The data that is recorded may indeed be produced and recorded by the internal monitors described in section II.3.1. Internal monitors provide "real-time" data, whereas the advantage of a log is that it is still available after the activity ends. The disadvantage is that normally much less is recorded than could be observed when the activity is in process. What logs are available varies from one machine to another.

Logs are typically stored on the computer executing the activity that is being logged. As in the case of the internal monitor, the machine on the causal chain is the one from which the tracker needs cooperation. The level of cooperation required to read logs varies with both the machine and the log to be read, similar to the cooperation needed to do different types of internal monitoring.

Unlike internal monitors, logs can be separated from the machine that performed the activity that was logged. In some cases logs are stored remotely. Then the tracker needs cooperation from the machine where the logs are stored. An administrator, at another site, might also be willing to share some log data with the tracker (because of pre-negotiated arrangements or because of an appropriate legal instrument) without granting any login permission. Also, unlike the situation with internal monitors, the tracker has time to convince the administrator to make log data available to the tracker after an incident.

II.3.3 Snapshot

A Snapshot is a complete or near complete copy of the state of a machine at some point in time, typically involving a copy of the disk, but ideally also including other state, such as volatile memory. A snapshot provides an amount of information similar to that available from internal monitoring, but, like a log, gives the tracker as much time as he needs to examine it. However, this information applies only to a single time. Like an internal monitor, the snapshot is of interest only for a machine on the causal chain.

Snapshots are typically used by law enforcement. The whole computer is seized and brought to the lab. Taking a snapshot is generally a very invasive procedure, requiring physical access to the computer. It also makes the computer unavailable for use, normally for an extended time. Removing the computer from service may be good for immediate defense but it is bad for attribution, since it affects upstream machines and may alert the attacker to the presence of the tracker.

It appears that the technology for snapshots is in need of improvement along at least two dimensions. One is recording contents of volatile memory. Another is the ability to do snapshots quickly, and allowing operation to resume without obvious impact. To the remote user it should ideally look like a momentary network problem.

II.3.4 Network Traffic

The tracker may be able to arrange to observe communication to and from the machines known or suspected to be in the causal chain. Similar capabilities are used for Level 1 attribution, so resources available for Level 1 attribution might be used for Level 2. Such observation requires zombie level control over the machines doing the observing, so the tracker needs cooperation from someone with that degree of control. An alternative to cooperation from someone who controls an existing machine is to put a new machine at a place where the traffic of interest can be observed. Installing a new machine requires cooperation of someone with physical control over the infrastructure where the new machine is to be placed. It also often involves cooperation on the part of those who control routers or switches in order to make the traffic of interest available to the observing machines.

Information about communication may enable the tracker to infer something about the activities of the communicating machines and the causal relationships among those activities. In the absence of extremely good knowledge of the machines involved (what programs are running and how they behave) those inferences are always uncertain.

The main advantage of network monitoring is that the tracker need not obtain the cooperation of the machine of interest in the causal chain. In order to observe the communication from one machine to another all that is needed is cooperation from *some* machine able to observe *some* link on the forwarding path. On the other hand, in order to observe *all* of the communication from or to one machine, one would need the cooperation of some machine very close to all of the network interfaces of that machine. Typically, the machine to be monitored has only one network interface and that is on a LAN. It then suffices to gain the cooperation of one machine on that LAN. (We use the term LAN here to mean a set of machines that can all observe all of the traffic sent to or from any of the machines in the set.) An alternative is to monitor the link that connects the LAN to the outside network. Typically there is only one such link for a LAN. More generally the tracker can try to collect a set of cooperating traffic monitors that form a perimeter around some set of machines, i.e., the monitors observe all traffic between that set of machines and the rest of the network.

An interesting combination of logging and network traffic would be logs of network traffic. We are not aware of any work using this data for Level 2 attribution, but we think that network logging is an important area for future work.

II.3.5 Reactions to Tracker or Defender Activity

Goal oriented behavior, including attacking, is generally improved by feedback. An obvious example is that an attacker who wishes to make a particular server unavailable would want to check whether the initiated attack is succeeding. If it succeeds initially and later the victim manages to take defensive actions that thwart the attack, the attacker would then try to figure out

how to overcome the defense. The activity of the attacker to determine that a defense is now working and to try to overcome it can be observed by the tracker. In this case the tracker uses whatever cooperation is already available to take actions that affect the attacker and observe what the attacker does in turn. The cooperation available determines what actions can be taken and what reactions can be observed.

The biggest advantage of the attacker with non-standard software level control (or better) is that the control may occur an arbitrary time before the resulting behavior. Therefore, in general, the tracker cannot expect to observe the controlling communication unless the tracker was fortuitously watching that machine when that communication occurred. This advantage can be overcome if the attacker reacts to a defense. For instance, the set of machines that successfully access the server between the time the defense starts working and the time the attack starts to adapt, very likely contains a machine controlled by the attacker and in contact with the attacker during that time interval. More precisely, unless the adaptation was pre-planned and actually independent of the defense, there must be a causal chain from the defense through the attacker (or at least some machine controlled by the attacker that decides how to react) and from there back to the attacking machines that changed behavior. We argue below (section II.5.2.1) that attackers who control many zombies can defeat this technique.

II.4 Survey of Existing Techniques

This section discusses how each type of data can be used to do Level 2 attribution for each category of control, along with techniques described in the literature. In general the tracker observes some behavior by some machine. The behavior may be observed from inside that machine or from traffic it sends. The tracker's objective is to determine whether that machine is controlled by another behavior in another machine, and if so which behavior and which machine. How the object can be achieved depends on the degree to which both the identified machine and the (at this point hypothetical) controlling machine cooperate with the attacker. Without a high level of cooperation from the already identified machine, it is generally hard to be sure of the attacker's level of control over that machine. It is even more difficult to know how much control the attacker has over the controlling machine, since it is not even identified yet. It is reasonable for the tracker to begin by asking what is the lowest level of control that is needed in order to explain the data that is available or can be easily obtained. Lower levels of control are easier for the attacker to obtain, and therefore more likely than higher levels. However, the tracker must also be wary of high levels of control being used to mislead the tracking effort. In each section below we describe what a tracker might do if there is suspicion that an identified machine is being controlled by an attacker at a given level.

II.4.1 Reflection Traceback

Level 2 attribution for reflection behavior is straight forward, at least in principle. A relatively small subset of all IP packets can be sent as reflections. Given such a packet, the first question is whether it was sent by the machine indicated in its source address. We call that machine the suspected reflector. Note that determining whether a machine is a suspected reflector is a Level 1 attribution question. If the packet was not sent by the suspected reflector then this is not actual reflection behavior, and the machine that did send the packet is spoofing the suspected reflector's address, and might be in the zombie class. If the reflector did send the packet then it is generally

easy to describe the packet that could have caused the reflection. The next question is whether any machine sent such a packet to the reflector just before the reflection was sent, and if so, which one. This is another Level 1 attribution problem. If no machine sent such a packet then the sender of the apparent reflection must be acting as a zombie.

The procedure above relies on Level 1 attribution, and this also depends on cooperating infrastructure. This is discussed in detail elsewhere, but a few remarks are specifically relevant here. First, the Level 1 questions above require attribution of packets that, in normal operation, would have been discarded before the question arises. Some of the Level 1 methods, such as packet marking, do not themselves save this data. Therefore, in order to answer such questions it will be necessary to add some infrastructure to save that data. The most likely candidates for this would be the suspected reflector itself or possibly a firewall or some perimeter of routers protecting that machine. Monitoring methods, by contrast, already store this data, and in that case the machines that cooperate in Level 1 attribution provide the data needed for Level 2 attribution.

In what we expect will be the normal case, where cooperation is not extremely widespread, Level 1 attribution is not very precise. We already know that Level 2 attribution of a reflection packet leads to either a legitimate behavior (the easy case) or to a zombie class machine. The fact that we are likely to not be able to uniquely identify that machine further complicates an already difficult Level 2 problem in the zombie case.

II.4.2 Stepping Stone Traceback

We begin with the classical stepping stone problem. That is, the tracker has reason to believe that an identified activity on an identified machine, M1, is controlled by some other machine M2, by means of a TCP connection following one of a small set of remote access protocols. If this hypothesis is correct then the methods below are meant to identify the connection by which M2 controls M1. The connection identifies M2 unless M2 is spoofing the source addresses of the controlling packets. If it is spoofing the source addresses, then identifying M2 is a Level 1 attribution problem. If M2 can be identified, the tracker will then attempt to determine how this activity on M2 is controlled.

II.4.2.1 Internal Monitor

If the tracker has enough access to a stepping stone machine to examine its activity, he can normally find the data needed for Level 2 attribution. For instance, in UNIX this data is available from the commands "*ps*" and "*netstat*". The first command identifies processes, the programs they are running and their parent processes. The second command displays network connections and the processes that use them. Stepping stone behavior involves a connection from the controlling machine to a server (e.g., a *telnet* server) running on the stepping stone, which creates a command line (shell) process, which executes the activity previously identified by the tracker. In a sequence of stepping stones that activity is normally another remote access program (e.g., *ssh*), with a connection to the next machine.

As a concrete example, the tracker may see that M1 is using TCP port 12345 to connect to the *ssh* port on machine M3. If the tracker can log into M1 with sufficient privileges, it is possible to use programs like *netstat* to determine which process is using this connection. This process

would normally be executing an *ssh* client. Its parent process could be found with a program that displays process information such as *ps*. That would normally be executing a shell program. If this is controlled by another remote access program from another machine, *ps* would indicate that some ancestor process of the shell process was running a server for that remote access program, e.g., a *telnet* demon. *Netstat* would then show that this process was communicating over a particular TCP connection, and identify the IP address and port on the other side of that connection, e.g., the IP address of M2, and port 23456.

II.4.2.2 Logs

Machines typically record remote logins, including useful data such as the IP address from which the connection originated. This data is not in itself enough to trace from an activity at one machine to a controlling activity at another, but it is useful for generating a set of candidate controlling machines, which in some cases may be very small. It is possible to record more data, as described in [Buchholz]. This sort of thing makes sense for an administrative domain that is interested in the ability to do such traceback within that domain, but of course it will not help for the usual case where the stepping stone is outside that domain.

The Honeynet Project [Honeynet] uses very extensive logging, which is even useful in the more difficult cases of non-standard software and zombies. However, the amount of data is too much to be practical for machines in normal use. Honeynet logging is described further below.

It is possible that the attacker might alter the logs. This behavior is beyond what would qualify as stepping stone activity. We therefore delay discussion of that problem until Section II.4.4., which deals with Zombie Traceback.

II.4.2.3 Snapshot

Snapshots provide the same sort of data described under internal monitors, so a snapshot should allow the tracker to identify M2. However, this method is much more expensive than internal monitoring in terms of access to the machine and, even worse, it disrupts the connection from M2. In particular, when M1 is stopped for the snapshot, M2 can no longer communicate with it. If M2 tries to communicate then this failure will cause the connection from M2 to M1 to terminate within a few minutes. If the tracker then identifies M2 from the snapshot data, access to the current state of M2 is no longer enough to follow the causal chain back from M2, since the controlling connection from M2 no longer exists. The access required for a snapshot is also sufficient for internal monitoring. Since internal monitoring is better than a snapshot in every way, there seems no good reason for using a snapshot in this case.

II.4.2.4 Network Traffic

The use of network traffic for stepping stone detection has received a fair amount of attention in the literature. In order to obtain the necessary data it is necessary to have the cooperation of a machine that can observe the traffic on the controlling TCP connection. If the stepping stone connections are unencrypted then content can be compared in order to match incoming and outgoing connections. [Staniford] is the earliest example of this technique. Even encrypted connections can be matched by comparing timing information, since stepping stone connections

tend to send data at irregular times, and the connections in a chain of stepping stones have high temporal correlation. [Zhang] is the first example of this technique.

It is worth noting that timing and data size characteristics of stepping stone traffic are preserved across not only a single stepping stone, but across a whole sequence of stepping stones. This fact means that it is not necessary to monitor traffic entering and leaving each stepping stone. If the tracker suspected that a chain of stepping stones originated in a given LAN, that hypothesis could be verified with only a single network monitor in the originating LAN and a single monitor at the victim, regardless of how many stepping stones are on the path between them. The verification would be done by recognizing the correlation between the traffic at the two streams at these places, and seeing no such correlation with another stream coming into the originating LAN. Of course, this identifies only two of the stepping stone machines, the first and last in the chain.

The direction of stepping stone control is actually implied by the TCP port numbers. The client has an arbitrary high port number, whereas the server has a low standard number, determined by the protocol, e.g., *ssh* is port 22, *telnet* is port 23, etc. If two different traffic monitors observe traffic along two different connections of a stepping stone sequence, the connections can be ordered by timing. This requires very good synchronization of the clocks at the monitors, however. The connection closer to the attacker (the "upstream" connection) sends from the client to the server before the downstream connection and sends from the server to the client after the downstream connection. Two traffic monitors that can observe traffic in both directions (from client to server and back) can be ordered without good clock synchronization, using the fact that TCP acknowledges every packet containing TCP stream content. The delay between a packet from client to server containing content and its acknowledgment from the server, as seen at an upstream connection, is contained in (hence shorter than) the delay as seen at a downstream connection. Similarly, if stream content is sent "upstream", from the server to the client, then the delay between data and acknowledgment at the upstream connection is contained in (hence shorter than) the delay at the downstream connection.

II.4.2.5 Reactions to Tracker or Defender Activity

[Wang01] describes a technique specifically for stepping stones without encryption. The victim sends extra data back to the attacker in a form that the attacker is expected not to notice. The extra data can, however, be recognized (if unencrypted) by cooperating machines on the path back to the attacker. This is beneficial in the case where the tracker uses network traffic to trace the attack, and the attack is noticed when the attacker still has an open connection but is not using it. Normally the tracker has to wait for traffic to be sent in order to see where it goes. This is a way to generate traffic for the tracker to observe without waiting for the attacker to generate that traffic.

II.4.2.6 Countermeasures

The tracking methods described above have been reasonably successful for the small set of command line interfaces listed, in the absence of countermeasures on the part of the attacker. Although encryption of the communication was not actually intended as a countermeasure, it did present a problem to be overcome. The solution was to use characteristics of *telnet*-like streams other than content, namely timing and packet length. Of course, these can also be obscured by an

attacker; for instance, by intentionally adding random delays to reduce the temporal correlation between stepping stone connections. There has been some work on what trackers can do in turn to overcome this sort of countermeasure. Examples include [Donoho] and [Wang03].

II.4.2.6.1 An Ideal Anonymizer

The two papers cited just above describe approaches that defeat particular kinds of timing perturbations. We now describe a general facility that attackers could use to prevent trackers from using any timing, content, bandwidth or packet length information. This facility therefore must defeat any tracking method that attempts to use those features. Facilities that prevent tracking are generally called anonymizers because they allow their clients to communicate without being identified. At most, the tracker may be able to see communication leaving the anonymizer and able to identify the set of clients. The tracker cannot determine which client is responsible for which communication leaving the anonymizer.

A set of anonymizing machines is organized into an overlay network. We assume that the anonymizing machines do not cooperate with trackers. The goal of the anonymizer is to thwart traffic analysis using network monitors. The network monitoring methods described above can trace back to the anonymization network, but after that the tracker can only tell that the control comes from one of the machines connected to the anonymization network.

The anonymizing machines communicate along established overlay links using encrypted traffic sent at a constant rate and restricted to packets of a single fixed size. All real traffic must be encoded into these streams. Excess capacity is simply filled with padding (which will still be encrypted.) Any client who wishes to use the anonymizing network establishes a connection with one of the machines in the network. This connection also uses fixed bandwidth, fixed length packets and encrypted communication.

The anonymization network and its clients will inevitably waste much of the fixed bandwidth that they use for communication. That is not a problem as long as the bandwidth used is a small percentage of that available along each link in use. Similarly, the service offered to an individual client will be limited to a relatively small bandwidth. This again is not a problem as long as that limit is above a minimum threshold required for usability.

At some point traffic leaves the anonymization network, e.g., to contact a server. This is not a fixed bandwidth, fixed packet size connection. The machine from which traffic leaves the anonymization network should be unrelated to the machine from which it enters, possibly a random machine in the network, or perhaps the one closest to the destination. The tracker can observe the delay between packets sent from the server into the anonymization network and their acknowledgments. If this delay is too short it can be used to eliminate distant anonymizer clients. Therefore the anonymization network should make sure that traffic is at least delayed long enough to prevent that analysis.

It is also desirable that all of the client machines connect to the network with the same bandwidth, or at least for any bandwidth in use should there be many clients. The reason is that if a web server is sending 1Mbps to the client then the tracker can eliminate as a suspect a machine that is connected to the anonymization network at only .5Mbps.

There are still some possible approaches available to the tracker. First, if the anonymizing network has a sufficiently small number of clients, it may be sufficient, for purposes of attribution, to simply think of all of these clients as suspects. It is advantageous for clients of the anonymization network to have a very large number of clients; for these clients to remain connected all the time (so the tracker cannot eliminate some due to the fact that they are disconnected when the controlling communication is observed); and, finally, that they all exhibit behavior that is indistinguishable to the tracker. It is also best for the clients if the vast majority of them are perceived as legitimate. That is, being a client should not be grounds for suspicion. Whether large numbers of legitimate clients can be recruited is largely a social issue. Such clients may join if they think they are protecting legitimate privacy concerns, or may not if they think they are protecting criminals. Laws requiring that anonymizers support traceback under certain circumstances may influence people to join or leave networks, depending on which governments they trust or like, and which networks are subject to the laws of which government.

One interesting approach we see for the tracker involves attacking the communication between the anonymizer and its clients. This is reminiscent of a Level 1 attribution technique described in [Burch]. If the tracker can disrupt that communication then, at least for the classical *telnet*-like applications, the communication from the anonymization network to the server will suffer highly correlated disruption. By waiting for the stream to the server to show activity and disrupting it for a number of varying but short periods at varying intervals the tracker can gain confidence that the stream he attacks controls the one he observes leaving the anonymization network.

On the other hand, the anonymization network could also detect the disruption. It could react by delaying *all* outgoing traffic by the same amount. This, of course, is bad for the anonymization clients. The entire anonymization network is very vulnerable to attack if killing a single client stops all traffic. Furthermore, this transforms random failures from minor inconveniences into widespread outages. This reliability problem can be reduced at the cost of some anonymity. Instead of every problem affecting all clients, clients can be put into groups, probably related to the bandwidth with which they connect and the reliability of their connections. The goal is then that no member of a group can be distinguished from any other. Over any interval comparable in size to the time it takes traffic to transit the anonymization network, the amount of data emitted for any member of a given group will be limited to the smallest amount received from any member of that group. For instance, a group might contain 100 machines that are all supposed to be sending 30Kbps to the anonymization network. If one of those fails to send any data for one second, then all of the output streams for all members of the group must also stop for one second.

Note that a similar method is always available to the tracker. If there is suspicion that some particular machine, M1, is transitively controlling another machine M2, the tracker can attack M1 and see whether the controlling behavior at M2 stops. The attempt on the part of the anonymization network above to cause that same effect when the tracker attacks another machine, M3 (which does not control M2), is the only countermeasure of which we are aware.

II.4.2.6.2 Existing Anonymizers

The idea of anonymizing software is surprisingly old. In work that anticipates the Internet [Baran](1964) describes a scheme that could reasonably be mapped onto the one above. For instance, this report says:

> "During the periods in which no valid traffic is being transmitted, a 'dummy' or filler stream of bits is sent, not only concealing traffic loading, but also for maintaining the timing synchronization."

It also proposes encryption both end to end and on each link.[12]

There is a variety of published work on anonymization. We describe a few examples very briefly here. [Onion Routing] does not prevent timing analysis but tries to limit the damage to the attacker should the tracker gain control over machines in the anonymization network. [NetCamo] seems to take into consideration all of our requirements above but in addition tries to satisfy real time quality of service requirements. (For typical stepping stone activity we did not expect this to be necessary.)

Anonymization software is already widely available[13], though not (yet) nearly as common as *ssh* or *telnet*. The market currently seems to be targeted more at anonymous web surfing than *telnet*-like applications. The products are meant to prevent a web server from identifying its true client, or finding identifying information about him. They do not protect against the sorts of methods described above. This is actually quite reasonable, since probably nobody actually has enough cooperation from machines around the Internet to trace back the single step from the anonymizer to its client!

II.4.3 Non-Standard Software Traceback

Beyond stepping stones the Level 2 literature becomes sparse. For tracing through machines running non-standard software it becomes useful to understand how the behavior in question is controlled. Such understanding was useful for the cases above as well, but since the software in those cases was assumed to be already identified and implementing a well-known standard, acquiring such understanding was not a problem.

II.4.3.1 Internal Monitor

Observation from inside a machine exhibiting non-standard software behavior allows the tracker to identify and examine the program that is causing the behavior. In principle, this allows the tracker to determine how that program works, though in practice it may be difficult. This solves one of the two problems introduced at the non-standard software level. The remaining problem of temporal lag between cause and effect can still prevent the tracker from identifying the controlling machine. The question that cannot be answered, in general, is how that program came to be running on this machine. Answering this question requires access to past activity, which is the job of logging. In some cases, of course, the program might be interactively controlled, which offers another path to a primary controlling host. In general, as in the cases above, finding the origin of the controlling communication is a Level 1 attribution problem. An

[12] One of the earliest references in [Daniels] is [Chaum] (CACM Feb. 1981), which describes an anonymizer for electronic mail. Chaum contained this interesting tidbit: "Baran has solved the traffic analysis problem for networks[1] but requires each participant to trust a common authority." He goes on to explain that the traffic analysis problem is the problem of keeping confidential who converses with whom and when. The reference was Rand Memo RM-3765-PR. Google leads right to it. See also [Baran-Bio]

[13] For examples see http://www.anonymizer.com , http://www.stealth-anonymizer.com/ , http://www.discount-evidence-eliminator.com/sw/anonymizer.htm , http://www.anonymizer.dk/.

intermediate case is what might be called occasional control, meaning that the program is listening for controlling communication, but an internal monitor cannot determine the origin of that communication until it arrives. If and when it does arrive, determining its origin is again, in the worst case, a Level 1 attribution problem.

II.4.3.2 Logs

In some cases non-standard software activity is sufficiently restricted that traditional logs can be very useful. In particular, if the attacker logged in at some point in order to retrieve and run the program generating the behavior to be attributed, normal logs would show such things as when the login occurred and from where. There may be many such events in the log, of course, but legitimate users can likely identify some of those, leaving the remainder as primary suspects. If the attacker can exert control without leaving evidence in the normal logs, of course, things become more difficult for the tracker.

[Buchholz], described under stepping stone logs, seems to be targeted at stepping stone activity, but not more general non-standard software activity. For example, that work does not attempt to record how a batch file was created. One might try to solve this problem by recording which processes create or modify which files. The attacker would then seek ways to cause processes created by others to perform tasks that aid attack objectives, such as copying and running files belonging to the attacker.

[The Honeynet Project] intentionally provides machines as targets for attackers in order to learn more about what attackers do. This uses a very aggressive form of logging, which is meant to allow researchers to understand non-standard software and zombie activity. By "aggressive" we mean that a large amount of very detailed data is recorded. This is not practical for normal machines that are intended for other productive work. It is justified in this case because the captured activity is almost certain to be malicious, and the goal of the research is to understand that activity. The current methods used are even capable of capturing plain text from encrypted communications in the normal case, e.g., where the attacker simply connects to the machine via *ssh*. However, using other non-standard programs could defeat this.

Although a small minority of machines in the network record enough data to be useful to trackers, even this small minority may be effective in identifying attackers who control large numbers of machines. The more machines an attacker controls, the more likely it is that some of them have the capabilities needed by the tracker. Currently honey pot machines do not provide what the tracker needs. They store a great deal of data for a short time. What the tracker needs is a summary of this data that is small enough to be stored for a long time and yet contains data that is useful for tracking. We think this is an important area for future work.

II.4.3.3 Snapshot

As in the case of stepping stones, the useful data available from a snapshot are also available at lower cost from an internal monitor. Again, a snapshot allows the tracker to determine how the current behavior is generated but that may not be enough to determine the previous machine in the causal chain. For that the tracker needs data from the time when the control was established.

II.4.3.4 Network Traffic

If the tracker understands the program running on the controlled machine then it is possible to know whether it is controlled from outside, and if so, how to recognize the controlling traffic. If it is not controlled from outside there will be no controlling network traffic to observe, so this data will not help the tracker. If there is active control from outside, then if and when controlling traffic arrives, the only remaining question is who sent it, which is a Level 1 attribution question.

Without the ability to do internal monitoring on the controlled machine it is difficult to be sure how the controlling program works. However, it is possible to make a very good guess in the case where the behavior matches that of programs that have been found in other places. The real value of honey pots is that they provide this data. Without knowing how the control works, the tracker can still make and check hypotheses. However, the attacker can easily supply misleading evidence. In particular, the non-standard software can start vast amounts of network communication, any of which could be used for control channels. The attacker might also arrange to change behavior shortly after some of these communication episodes. All of this gives the tracker leads and work to do, but may not result in actual benefits.

II.4.3.5 Reactions to Tracker or Defender Activity

Another technique that can be used by the tracker is to assess any reactions to the tracking activity. The object here is not to provoke a reaction from the machine that the tracker has already identified, but to provoke one from the machine that controls it, in hope of identifying that machine. It is possible that the tracker might know something about how the controlling machine works even though it has not been identified. The most likely way for the tracker to know this is to have found other pairs of controlling and controlled machines exhibiting the same behavior and to know that the programs came from those machines. In that case, the tracker would know what, if anything, can be done to cause the controlling machine to communicate with the controlled machine. Knowing this information, the tracker can prepare to observe the controlling traffic and use Level 1 attribution methods to identify the controlling machine.

If the tracker does not know how the controlling machine works, but does know how the controlled machine works, it might still be possible to guess about actions that might cause such communication. This is likely to result in the controlling communication arriving sooner, and thereby speed up the attribution process. Factors that would influence the guesses are: what has worked in similar cases before; the effects of attacker activity *(what goal does it appear to accomplish)*; and how would various tracker actions affect that.

If the tracker does not even know how the controlled machine works, while guesses similar to the above can be made, the tracker also has to guess about the controlling communication. If the controlled behavior changes in response to tracker activity, then there must be some communication path to the controlled machine that causes the changes, and that communication must arrive in time between the tracker action and the reaction on the part of the controlled machine. The causal chain leading to the change in activity starts with the tracker activity and leads to the change in activity of the known controlled machine. It may be easier to find the attacker (or some upstream machine under the attacker's control) by following the causal chain forward from the tracker activity than to try to follow the causal chain back from the controlled machine.

Tracker activity that the attacker could have foreseen might cause reaction on the part of a machine without direct attacker intervention. If the tracker can do something that the attacker does not expect then any reaction will have to follow a causal chain directly through the attacker, which, in terms of Level 2 attribution, means that it goes through a primary controlling machine. (One might argue that the attacker's reaction was "caused" by the tracker's action, but our notion of causality was intended to be limited to automated agents. Therefore, our view is that one causal chain starts with the tracker and ends with something observed by the attacker, and another begins with the attacker's reaction.) The reaction might follow a different causal chain from the attacker to the victim than the original attack, but all such causal chains (and especially primary controlling machines) are interesting as Level 2 attribution results. If there are multiple attackers then all attackers are also interesting as Level 3 results.

II.4.4 Zombie Traceback

II.4.4.1 Internal Monitor

In the non-standard software case the tracker with access to internal monitoring could determine what program was controlling the machine and how it worked, but could not determine (without extraordinary logging) which machine to blame for that program running. With zombies the tracker loses even the ability to determine what program is controlling the machine. An attacker with zombie level control over a machine can allow login access to the tracker and present to the tracker with any view desired by the attacker. The attacker can even make it appear that the tracker has zombie level control. Interesting examples of misleading views that could be presented include:

- Nothing going on here
  There is no trace of the controlling activity of interest. The tracker may even detect traffic on the LAN but not be able to tell by internal monitoring which machine is sending it.[14]
- This machine is controlled by another machine
  For instance, this machine might be running non-standard software that is clearly generating the activity of interest. However, that program explicitly looks for direction from a certain other machine. The other machine may, in reality, be innocent. The attacker would probably misdirect the tracker to a machine that will present difficulties in gaining access, or perhaps even a machine that that the attacker wishes to target for an attack, in hope that the tracker will be able to obtain a warrant to seize that machine.
- This is the primary controlling machine
  This would be most useful in the case where the attacker knows that the machine will be inaccessible to the tracker, in order to thwart further tracking efforts.

Deception as above used to be theoretically possible but difficult in practice. The availability of such software as VMware and Virtual PC now makes such deception relatively straightforward to implement. It is, of course, possible for the attacker to make any number of mistakes that will

[14] This is a Level 1 attribution problem which is generally beyond the scope of what is addressed in our Level 1 paper. Physical properties of the signal can be used to determine which machine is the sender. In general, physical access to a machine would allow the tracker to determine which packets it sends.

help the tracker. The point here is that the attacker has a fundamental advantage in this case. It is worth mentioning that this same advantage works to the advantage of the tracker in the case of honey pots. In that case, the tracker has zombie level control and allows the attacker to think that the attacker has managed to gain zombie level control. Both cases lead to a complicated game in which the first party to control a machine tries to convince the second party that there was no earlier control. Meanwhile, the later party to control the machine tries to recognize whether or not there actually is an earlier party who is now able to observe all the behavior at that machine!

II.4.4.2 Logs

Logs have the same problem as internal monitors: the attacker controls the data in the logs. However, there is one interesting exception. It is possible to arrange for log data to be recorded in a way that cannot be altered after it is written. The attacker would then control logging after gaining control, but the activity involved in gaining control would still be visible to the tracker. Ideally this log data would identify the machine(s) used to gain control. This does not help the tracker to determine how this zombie is communicating, but it is fairly strong evidence that the same attacker controls both machines, and that the other machine is "upstream", i.e., fewer control connections to the attacker. This in turn is reason to switch attribution focus to that other machine. Possible flaws in this reasoning include:

- The controlled machine might have been taken over again since the attack in the log, in which case the log would lead to an earlier attacker rather than the one who now controls the behavior of interest.

- The machine identified in the log might have been taken over since the incident in the log (including the possibility that the original attack was discovered and now the rightful owner has reestablished control, or even the possibility that the attacker intentionally returned control, removing any changes made to the machine). In this case the log does point to a machine that was controlled by the current attacker, but techniques that determine who now controls that machine lead away from the correct causal chain.

An attacker who is worried about such logging might reasonably try to exploit these problems. The attacker would use one zombie (which does not do such logging) to take over other machines and turn off their logging. Then another zombie can be used to control those same machines. Finally, the first zombie is used to remove all traces of itself and return control over its hardware to its rightful owner, or even leave evidence to frame someone else (the real victim).

II.4.4.3 Snapshot

A snapshot is the only theoretically effective solution to the problem described under internal monitors (and affecting logs other than cases where the logs are unmodifiable). This brings the tracker back to the same position as in the non-standard software case. The tracker can figure out how the machine now behaves, but cannot reconstruct enough history to determine what machine was used to put it in that state. The term "theoretically effective" is meant to convey first, that it is not easy (and nowhere near automated), and second, that with enough effort it can be done. The word "only" is meant to convey that, while other methods might work in some cases, this is the only one with enough data to work in every case. This assumes, of course, that

the attacker does not detect the tracker in time to destroy that data. A more realistic problem is that the tracking method itself will alert the attacker to the presence of the tracker. While the tracker is figuring out how this zombie works, the attacker will arrange for no other zombies to try to control it in the future, and perhaps just to be on the safe side, cleanse the zombie that controlled it of all data that could lead further back along the causal chain. Then, if he suspects that the tracker will figure out how this zombie works, the attacker might change all of the zombies that work in the same way to work differently.

II.4.4.4 Network Traffic

For zombies, as for non-standard software, it is possible to make effective use of network traffic when the machine was actively controlled from outside and the program controlling the machine was well understood. In general, however it is more difficult to understand the program controlling a zombie. In the worst case the program can be recovered only by analysis of a snapshot.

Zombie level control also introduces the ability to send and receive arbitrary packets, and this is the reason that the tracker may need a general Level 1 attribution facility. Fortunately for the tracker, it is not particularly easy to communicate with IP addresses other than those of the communicating machines. Unfortunately, it can be done. This actually seem to be getting more difficult as the general trend in networks tends toward reducing the amount of traffic that normal hosts can observe, e.g., hubs tend to be replaced by switches.

II.4.4.5 Reactions to Tracker or Defender Activity

This appears to be the same as for the non-standard software case.

II.4.5 Physical Traceback

We argue above that, if the attacker does have physical control over a machine, the attacker would like to keep that information from the tracker. The attacker will avoid any activity that would implies physical control, especially if the tracker might be able to detect that activity.

II.5 Summary of the Current State of the Art

This section presents an overview and discussion of section II.4. In particular, we want the reader to see the boundary between the regions where Level 2 attribution can be done and where it cannot, and to understand what happens at that boundary that makes the difference.

II.5.1 Summary table

We now present a table summarizing section II.4 above. We use "+" to indicate the presence of techniques that use the given data and work reliably given a machine controlled in a given way to find some upstream controlling machine if there is one, "–" to indicate absence of such methods, and blank to indicate that Level 1 attribution is sufficient (which we assume is available for use in all these cases).

Table II.1: **Summary table**

data source	reflector	stepping stone	non-standard software	zombie/physical
internal monitor		+	– [1]	–
logs		+ [2]	+ [2]	+ [2][3]
snapshot		+ [4]	– [1][4]	– [1]
network traffic	[5]	+ [6]	–	–
reaction		+ [6]	– [7]	– [7]

Notes for Table II.1:

[1] This is useful for determining how a program controls behavior and for determining the source address of any ongoing outside control, but is not sufficient for attribution in the absence of controlling communication.

[2] Existing logs are generally not sufficient but useful for at least narrowing the possibilities. Additional logging is possible and useful.

[3] The logs must resist alteration, e.g., by being recorded on another machine or on write-once media.

[4] This is as effective as internal monitor but higher cost.

[5] The ability to observe network traffic is needed for Level 1 attribution.

[6] This works in the absence of strong anonymization.

[7] One problem is that the attacker cannot in general be forced to react. Another problem is described below.

II.5.2 Discussion

The table suggests that Level 2 attribution is feasible for reflection and stepping stone activity. It should be noted that this still relies on the ability to obtain some of the data, which requires cooperation from some machines on or near the causal chain. This is already an insurmountable obstacle in many cases. On the other hand, the table suggests that there is no hope for non-standard software level control, zombie control, or physical control, short of preserving historical data far in excess of what is normally kept today. Today this would be something of an overstatement. A more accurate statement would be that a tracker who identifies a machine M at some time T, even with the ability to examine the program controlling M and the ability to observe all communication to and from M after T, cannot in general determine whether the control of M was established from some other machine, and if so which one. The main problem is that the attacker can cause a machine to act under control of a program at a later time, leaving no information on that machine to indicate the origin of that program. In this case the only hope for the tracker is data collected before time T, i.e., before the tracker realized that M was involved in an interesting causal chain.

II.5.2.1 Limitations on Reaction to Tracker Activity

We have suggested that reaction to tracker activity may be a powerful tool against the problem of arbitrary time lag between control and the activity it causes. In particular, it limits the search for a causal chain to the time interval between the tracker activity and the attacker reaction. The implicit assumption is that it is sufficient to trace one path back to one primary controlling host. Although the tracker does not have sufficient historical data to determine how a program now running came to be running, it is possible instead to observe the communication that causes the program to change behavior and trace back along that path. The tracker actually has two approaches. One is to try to follow the causal chain forward from the tracker activity to the attacker, who must at least detect the effect of the tracker activity in order to react. The other is to trace back from the machines that react. We argue that the attacker can prevent both approaches from working.

As an example, suppose the attacker is making a particular web site unavailable by flooding it. The tracker might respond at time T1 with a DNS update so that legitimate users go to a different IP address. If the attack changes at time T2 to attack the new address then there must be some causal chain from the new DNS entry to the new attack. That chain must start from a DNS lookup. For simplicity, suppose there was only one such lookup in the interval from T1 to T2. This could, of course, be from an attacking machine checking and reacting on its own. Or it could be from another machine controlled by the attacker, which has been preprogrammed to check and then instruct the attacking machines to alter their target addresses. We will assume, instead, the best case for the tracker: this machine communicates through some number of steps to the attacker, who then communicates through some number of steps to the attacking machines.

Our first argument is that the attacker can make it very difficult for the tracker even in this best case for the tracker. The problem is that the sensing machine can communicate with a very large number of machines, and these might communicate with many more machines. Even a complete understanding of the code running on the sensing machine does not help the tracker. What differentiates the attacker from all of the other receiving machines is what it does with that communication, and this cannot be determined from the sending machine. For instance, the tracker might see that the sensing machine (the one that did the DNS lookup) sends a ping packet containing the new IP address to attack. But it can send similar ping packets to 10000 different IP addresses. Alternatively, the sensor might send the new address (perhaps hidden in a message that looks like spam) to a very popular mailing list or newsgroup where it will be read by thousands of machines in the next hour. The attacker might carry out several such steps to impede the progress of the tracker.

We next consider what the attacker can do to prevent the tracker from tracing back from the updated attack. Suppose, for simplicity, that only one machine, M1, was attacking, and at time T2 it changed its attack to the new IP address. This tells the tracker that one of the machines that communicated with M1 between times T1 and T2 was controlled by the attacker. The value of understanding the code controlling M1 was that the tracker could determine which of the two machines was under the control of the attacker. However, even this might not help in finding the primary controlling machine. If the attacker can establish zombie level control over a large set of machines, such machines could be used one time only in subsequent control.

To see how this works, suppose the attacker controls M1, M2, M3, etc. The attacker arranges for M1 at some future time T1 to start attacking. The attacker also arranges for M1 to accept control of a special message, which can be received from any machine at all. Finally, before time T1, M1 will erase all evidence of the origin of this control. The tracker sees the attack at time T1, determines that it comes from M1, determines how M1 works, prepares to detect the controlling communication, and takes a defensive action. The attacker senses the defense, which we have shown above can be done without significant danger of identification by the tracker. Now the attacker decides to react. The attacker arranges for M2 to tell M1 how to react at time T2, and before T2 to erase all evidence of the origin of this control. Now at time T2 the tracker sees this communication and concludes that M2 controlled M1. This is true. But the tracker is still no closer to the primary controlling machine! Every time the tracker causes the attacker to react the attacker simply uses another zombie. In the current Internet there is apparently no shortage of zombies. The key point is that the tracker needs access to historical data even in the case where the attacker does react. In the example above, the tracker needs that data to determine who caused M2 to send the controlling data to M1[15].

The point here is that causing the attacker to react does not remove the need for data about the activity of a machine before it could be recognized to be on the causal chain. On the other hand, the attacker did control M2 after T1 in the example above. So it would be sufficient for the tracker to record all of the activity of all machines that could possibly be on the causal chain starting at T1. This does not sound very practical, but it is at least easier than recording all of that activity from even earlier, which is what would be required if the attacker cannot be made to react.

[15] Of course, the attacker might just as well let M1 continue to attack the old address and let M2 attack the new one. However, if there were 100 attacking machines, the attacker might not want to use up 100 new zombies to react to one action. The attacker would rather use up one new zombie to tell all of the others how to react. Then, perhaps the new zombie would simply join the attack. The tracker's actions then actually make the attack incrementally worse!

II.5.3 Current Attribution Process

Our goals in presenting the process are, first, to show the reader the overall Level 2 attribution picture from a tracker's perspective, and, second, to point out the gaps in what is possible either because of non-cooperation or lack of technology or both. This section describes what a tracker can do to attribute observed activity in the current Internet. Of course, the Internet is the worst case. In a private network the attribution problem might be much simpler. However, the Internet is also the most common case, and therefore the case of most interest.

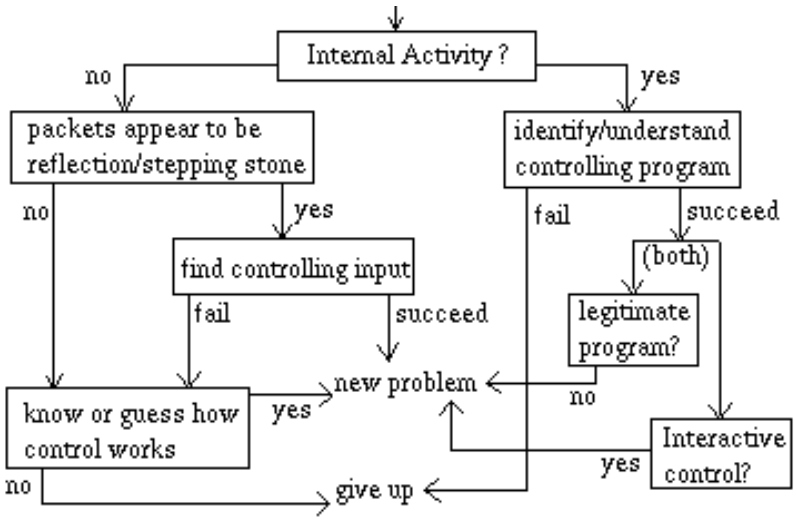

Figure II.3 **Single Iteration of Attribution Process**

Figure II.3 is a high level flow chart showing a single iteration of the process. The iteration begins with a goal of finding a cause for an identified activity. It ends either with the identification of such a cause or the failure to do so. The successful identification appears in the diagram as “new problem”, since the tracker now has a new problem of identifying the cause of the newly identified activity. A failure to identify a cause appears in the diagram as “give up”. This failure could be due to the fact that the identified machine is a primary controlling host. It is more often due to the fact that the tracker simply does not have the data needed to find the cause. Many details are left out of the diagram and described in the text below.

A Level 2 attribution problem starts with an identified activity on an identified machine. Which activities can be identified depends on whether the tracker has internal access to the identified machine. Without internal access to the machine, the only activity the tracker can observe from the machine is the packets that it sends, so the identified behavior must be described in terms of those packets. If the tracker does have internal access to the machine then it is possible to observe the packets, but the identified activity could also be described in terms of programs running on that machine.

The Level 2 problem sometimes requires attribution of past activity. In the current Internet, the data required for past activity is even less likely to be available than the data required for current activity. We view as a gap in current technology the fact that even the most cooperative

machines are unable to supply historical data that would allow a tracker to solve any but the most trivial Level 2 problems.

- "no" branch of "*internal activity*" test:
  The activity here is described in terms of packets sent. The tracker can presumably observe at least the packets in question and can easily tell whether they are consistent with the hypotheses that they are caused by reflection or stepping stone activity (without anonymization).
- "yes" branch of "*packets appear to be reflection/stepping stone*" test:
  The hypotheses that the packets result from reflection or stepping stone activity are relatively easy to check if the tracker can observe all of the traffic sent to the identified machine. If the tracker can observe not all but at least some of the traffic sent to that machine then it is possible to check whether any of the machines sending the observable traffic is controlling the identified machine. If the tracker does find incoming traffic that seems to be causing the identified activity then the tracker has probably (see below) succeeded in taking one step back along the causal chain, and the result is a new attribution problem to determine how the incoming traffic is controlled.

  More often, the tracker does not have the cooperation needed to observe this communication. Therefore, the tracker cannot tell whether the attacker is really using a higher level of control. The tracker is almost certainly stuck either way, i.e., there is no further to go back in the causal chain. (Note that internal access, if available, is a more direct way to find machines controlling stepping stone activity, but internal access is normally sufficient to observe all communication, so the tracker who cannot observe the controlling traffic will not have this option.) Since stepping stone control is easy for the attacker and the tracker is unlikely to have enough cooperation to trace back through even one stepping stone, it makes sense for the attacker to use simple stepping stones for control in the current Internet, and therefore it is very likely that what appears to be simple stepping stone activity actually is.

  It is possible that an attacker could make it appear that a machine is controlled as a stepping stone (even if it is actually controlled through some other means or is even the primary controlling host). For instance, the machine could send packets to itself with spoofed source addresses and even spoofed MAC headers to indicate that they were forwarded from the local firewall. It is a Level 1 attribution problem to detect the true source of those packets. In practice, Level 1 problems are also difficult to solve in the current Internet, again due to lack of cooperation.

  Another possible attack that is more difficult for the attacker (and therefore less likely to be used) is for the attacker to find an innocent machine that has opened a connection with the identified machine and is sending and receiving encrypted data. The attacker could adjust the outgoing traffic identified by the tracker to correlate well with the innocent traffic. This scheme would be more useful for communication if the innocent traffic were known to maintain a minimal useful bandwidth. (A connection to an anonymizer as described in section II.4.2.6.1 would make a good

candidate.) In the worst case, the tracker needs something similar to a snapshot of the identified machine in order to be sure of the result. The fact that a snapshot may be required to determine whether a machine is actually cooperating with the tracker or attacker is not a symptom of either lack of technology or lack of cooperation. We view it instead as a fundamental feature of general purpose computers that they can be programmed to mislead, and the party with stronger control has the power to supply misleading data to the party with weaker control.

- "no" branch of "*packets appear to be reflection/stepping stone*" test:
  The alternative to stepping stone or reflection activity is non-standard software or higher degree of control. The case where the identified machine is the primary controlling host falls under physical control. The tracker now has the problem of understanding the controlling program. If the identified machine cooperates with the tracker (very unlikely in the current Internet), the internal access to that machine can be used to recover and analyze the controlling program. If internal access can be obtained then the situation is changed to the right side of the figure. Otherwise the tracker might recognize the behavior from past experience and know (with some limited confidence) how the controlling program works. This method would be more likely to succeed and the results would be more reliable if trackers cooperated more in ways that allow the experience of one to benefit others. One of our goals is to promote this kind of cooperation among trackers.

  It is also possible to guess how the controlling program works, but guessing is, of course highly unreliable. For instance, a correlation between outgoing communication and ping packets observed on the local network might be considered strong enough evidence to justify further attribution attempts starting from the ping packets. However, skepticism is suggested by the fact that the data on which guessing relies tends to be under the control of the attacker. The most likely outcome is that the tracker has to give up because none of the previous methods produces an understanding of the controlling program. If recognition or guessing is considered promising, then the situation is similar to that on the right side of the diagram, which is described below. However, the tracker is less likely to succeed because the internal data that is normally available on the right side of the diagram is lacking.

- "yes" branch of "*internal activity*" test:
  If the activity is described in terms of programs running on the identified machine then the tracker presumably has internal access to that machine, which can be used to identify and analyze the program generating the identified activity. Again, an attacker with a high degree of control over the identified machine can present misleading data to the tracker, and in the worst case, the tracker needs something like a snapshot in order to be sure of his conclusions.

  Normally the tracker can identify a program that generates the identified activity. He can, if necessary, examine that program to see how it works. It may be a standard program, which is already understood or at least believed to be understood by the tracker. The tracker may want to verivy that the belief is true, and that the program is not an altered version planted by the attacker. The tracker may also see that the identified program is controlled by another program or is controlled by network

communication. Further, the tracker can identify the controlling program or communication and then consider the new attribution problem of finding the source of that controlling communication or program.

The other potential question for the tracker is how the identified program came to be running. The program may be known to be legitimate and to have been legitimately run, e.g., a web serving program that is supposed to run on this machine. If, however, the program is not thought to be legitimate, then the tracker has new Level 2 attribution problems: *where did this program come from and how did it get started*? This new problem is an example of an attribution question about past activity. Usually, the evidence is insufficient and the tracker has to give up. In the best case, there might be logs showing that an intruder logged in at a given time from a given IP address and perhaps even copied the program in question from another identified machine. These pieces of evidence lead the tracker to some further attribution problems. Unfortunately, these are very likely to be unsolvable for two reasons. First the tracker is unlikely to have access to the identified machines. Second, even if such access is obtained, those machines are unlikely to have recorded enough history to allow the tracker to track back further.

There are generally two possible results of the activity above. The "successful" outcome is a new problem starting from a point on the causal chain that is closer to a primary controlling host. In order to be sure that the new starting point is actually on the causal chain the tracker might need to examine a snapshot. The "unsuccessful" case could represent a dead end, but it might also represent a primary controlling host. It is sometimes possible to find evidence that a given host is primary, such as observation of stepping stone activity from the host without any incoming control. It is certainly possible that an attacker could record an interactive session and arranges to replay it from a zombie planted on an innocent machine. However, some activities are currently beyond the capability of programs. An example would be a chat session in which the tracker participates and the attacker responds in a meaningful way. This example is an instance of the tracker provoking a reaction from the attacker, but the value lies in showing that an already identified host is primary rather than in identifying the next step on the causal chain.

If the tracker can gain access to a suspected primary controlling host, the machine might be found to contain incriminating evidence. The tracker might also determine where that host is and who has physical control over it. There may be reason to suspect that person. The tracker, in some cases, might be able to arrest that person and seize the host. Of course, success is the exception. The rule is that the tracker has no such evidence and simply has to give up.

In summary, the "failure" outcome is highly probable in the current Internet. Most often the cause is lack of data due to lack of cooperation. Longer causal chains require the tracker to traverse the flow chart many times, and this increases the chance to fail before finding a primary controlling host. On the other hand, multiple paths of control increase the chance of finding at least one primary controlling host. There is much that the attacker could do to impede the tracker's progress, but in the current Internet there is little need for the attacker to take such measures. If trackers can gain better access to data, then some of the other issues we have described above, such as historical data, can be expected to become more important.

Chapter III

Techniques for Level 3 Attack Attribution [1]

III.1. Problem Statement

The Level 3 attribution problem is to identify the individuals responsible for an "attack" based on evidence observed in the state or activity of one or more computers that are connected to a network. At the very least, examples of such evidence include files on the disk of a computer, programs running on a computer, and network packets involving the computer or the network of interest. The Level 3 attribution problem, however, is difficult separate from the context of incident response and forensics investigation in which it is situated. Some of these contextual points are examined more carefully now.

In this report, the person(s) doing the attribution is called the "tracker(s)". The term "attacker" means anyone whom the tracker is trying to identify, and anyone else who may try to hinder the efforts of tracker in doing so. This project is concerned only with attribution of activities that involve the use of computing and network resources.

In Level 3 attribution, as with Level 1 and Level 2 attribution, there is no need for or value of a restriction to attribute "attacking" behavior, as opposed to behavior of any other kind. The data and techniques necessary to attribute attacks are no different from those required to attribute other behavior. So, the purpose of Level 3 attribution is to uncover causal relationships between observed data (indicative of some behavior or activity) and the human actor(s) responsible for that behavior or activity. By way of further clarification, within the scenarios of "attack" behavior, this research is concerned with behavior in which the computer or network might be the eventual target of the attack (e.g., installation of malware on a computer). In addition, there is also interest in behavior where the computer or network is simply being used to commit other crimes (e.g., stealing private information or documents).

Ideally, the Level 3 attribution process would actually identify the individual(s) responsible for an activity. Unfortunately, most computers in current use have very limited ability to sense human interaction other than through a keyboard or mouse. Moreover, such interactions are not normally stored. Therefore, there is usually not enough information to link the limited data to the specific individuals associated with producing that data in a fully automated manner.

The Level 3 attribution problem has its roots in traditional law enforcement and forensics processes, which invariably include a human tracker evaluating evidence or clues and constructing a logical chain of explanations leading to the human perpetrator of an observed activity. In real life, the forensics process is full of heuristics and insights learned from experience. Human expertise is still required for successful forensics. Thus, rather than use an "all automation or nothing" approach to Level 3 attribution, this project hopes to enhance both the ability of the tracker to gather information and analyze it within a real-world forensics context. Moreover, the project is predicated upon a broader definition of the Level 3 attribution problem that allows for the tracker to build up a profile of the individual(s) responsible for an

observed activity, or determining whether the same or different individuals are responsible for two different activities.

This research considers only IP networks paper because this is the primary protocol for much of the world's networks. Where necessary, any ideas developed during this project can be extended in fairly straightforward fashion to non-IP networks. Finally, while it will be necessary for the tracker to be able to integrate many different data sources and correlate them in order to be successful in achieving Level 3 attribution, as it relates to actually supporting the gathering of evidence, the research is confined to information that can be gathered through computers and networks. Such observable data or evidence is defined as the "*measurables*" for the remainder of this document.

Relationship to Level 2 Attribution

Level 2 attribution has as its goal the identification of the "primary" controlling computer(s) in the *causal chain of machines* that control the computer that is observed to be attacking a victim site. There is an implication in the BAA that Level 3 attribution is strongly related to (and perhaps even dependent upon) the results of successful Level 2 attribution. Using the definition of Level 2 attribution as above, the humans responsible for the activity being attributed must be the direct, local users of such primary machine(s) identified via Level 2 attribution. In the era of the "personal" computer, where most machines are typically used by a very small number of people, successful Level 2 attribution gets the tracker quite close to successful Level 3 attribution.

There are cases where one can perform Level 3 attribution without performing Level 1 or Level 2 attribution. For example, in some attack scenarios, at a victim site one might have access to attack code that can be analyzed to identify (with a high probability) some of the author's characteristics without knowing either the attacking computer (Level 1 attribution) or the controlling computer(s) (Level 2 attribution). This project will look at both cases where Level 2 attribution results are available (and possibly necessary) to aid Level 3 attribution, and cases where Level 2 attribution is not possible or not necessary.

It must be noted that Level 2 attribution always has some associated uncertainty as to whether a particular machine is "primary" or just another machine in the causal chain of machines causing a particular activity. Level 3 information might aid Level 2 attribution in that regard by providing some evidence about the person(s) using the primary machine. If Level 2 attribution results have a high degree of certainty (thanks to other information that the tracker has at his disposal), then that could aid Level 3 attribution by providing a basis to exclude those users who could not possibly have used those machines.

Another point to note here is that identification of *where* cooperation is needed in the network to perform Level 3 attribution is largely determined by the Level 2 attribution process. The primary controlling machine and, possibly to a lesser degree, the other machines on the Level 2 causal chain have access to the Level 3 data needed. From the tracker's perspective, in the worst case, all he has access to is data at the victim site. In the best case, from the tracker's perspective, he has control of the primary Level 2 machine, and can gain access to all the data there, which might be used for Level 3 attribution. In general, even if one cannot trace back to the primary

controlling machine, it may do considerable good to find (and control) a machine further up the causal chain because that machine could well provide additional, valuable Level 3 data.

III.2 Framework to Describe Related Work

The goal of this document is not to describe every research project that might somehow be related to this effort. Rather, the goal is to provide an overall feeling for the coverage of the human characteristics that is provided by focusing on different measurables and how complicated (from a computing aspect) the analysis methods that work with each kind of data might be.

There is a large volume of work (from computer forensics to language analysis to biometric analysis to good old-fashioned detective work) that one can view as "related" to the problem of Level 3 attack attribution. In order to avoid getting bogged down in "apples and oranges" comparisons of methods that have very different goals and perspectives, the related works are assessed using a common framework that covers several (non-exclusive) aspects.

III.2.1 Assessment Dimensions

The following different dimensions are used to initially describe and later assess the related research and work:

Measurables – Different existing methods are predicated upon different kinds of data. The biggest assumption underlying any Level 3 attribution method is the availability of the data required for that method. This is an important issue because, in practice, Level 3 attribution will likely be performed in various environments where very different data will be available (or unavailable) at different points of the network (closer to the attacker or closer to the victim along the Level 2 causal chain). Examples of measurables include documents, email, attack code, and keystroke timings. Different measurables have the potential to uncover different characteristics of the human actors who are responsible for producing them.

Whether or not a specific measurable is available is also a matter of getting cooperation from different parts of the infrastructure, which is another important concern for our research project. In a contemporary Internet-based attack scenario, much more data will be available to the tracker at or near the victim's site than will be available closer to the attacker along the Level 2 causal chain[16]. Since the objective of Level 3 attribution is to identify some of the characteristics of the attacker, however, in some cases, data nearer the attacker could prove to be "richer" or more relevant for Level 3 attribution. Cooperation is presumably much easier to come by within one's own infrastructure (or closer to it) than at remote locations. Thus, a focus on the measurables becomes important in assessing whether a particular technique that uses that kind of data is realistic or not in a particular situation.

Techniques to Analyze Each Measurable – in each of the different cases, different analytical techniques are used with the data at hand. These techniques are usually adaptations of more general mathematical or statistical analysis. It is useful to understand the full range of such analytical techniques that have been used. Note that the same analysis technique maybe used

[16] In a typical attack scenario the tracker works with the full cooperation of the victim.

with very different data, and that different researchers may employ distinct analytical methods on the same data. In each case, however, the basis by which the technique can be used to perform Level 3 attribution is described.

Techniques for Integrating Multiple Measurables – one might liken the job of performing Level 3 attribution to putting together a composite description of a suspect from the perspectives of different witnesses. The paradigm of integrating multiple perspectives into a single composite description is particularly useful in Level 3 attribution because of the diverse set of measurables and techniques involved and the fact that the characteristics deducible from a particular analysis are usually disjoint from the characteristics deducible from another. This research specifically places focus on techniques that are devoted not only to a particular measurable but also to creating a composite description that integrates multiple measurables and multiple techniques. This set of methods to integrate perspectives might be disjoint with (yet complementary to) the set of methods that analyze a single kind of data (e.g., email messages) to find the characteristics of the human actors involved. For both research and presentation clarity, it is useful to distinguish between the techniques that provide attribution from a single perspective and the techniques that are capable of integrating multiple attribution perspectives.

III.2.2 How Data Are Gathered

Level 3 attribution methods require data. Data can be gathered at the victim site or outside the victim's site. There are known ways in which such data can be gathered. Each technique to gather data requires the establishment of either cooperation with or some level of control over the computers about which information is desired.

Network Traffic Monitoring

The packets that constitute network traffic are among the most commonly available measurable for attribution. The kinds of information about packets that are expected to be used are as follows:

- *Packet Headers*: between the IP, TCP, MAC headers, a lot of information can be gained such as the nominal source address, destination address, routing data, whether or not the attacker is local, operating system fingerprinting data, nature of the infrastructure on the other side of the communication, and so on.
- *Packet Timing*: can reveal information about the infrastructure, about the program sending the data, and the individual providing input to the program.
- *Packet Content*: shows information about the software being used and also indicates some characteristics of the individual involved.

Some of the data above is controlled by the attacker, and can be falsified ("spoofed") in order to mislead the tracker. However, some of the data (such as TTL) is not fully under the control of the attacker. The attribution techniques the tracker uses must ideally work even in the presence of deliberate attempts to misdirect or deceive the tracker.

One can certainly observe ALL the traffic within one's own network. With administrator privileges, one can use commands like "*tcpdump*" or its equivalents to see network traffic activity at a packet level. At its simplest, packet analysis can provide a picture of who is communicating with whom, and the volume of such communication. Further, if the

organization's policies allow it and if there is no encryption being used, it is conceivable that one can examine the contents of packets as well.

Outside one's own network, typically the ability to gather information is considerably limited. However, information that is away from the victim and closer to the attacker could prove to be very useful. In some cases, data at the victim might be enough to provide Level 3 clues – such as the level of knowledge of the attacker. In other cases, however, it might help to be able to go one hop closer to the attacker (in the Level 2 causal chain) could prove to be very useful in providing information about the type of attack and method of control used. Unfortunately, it is difficult to get the data upstream from the victim without cooperation, as described in the Level 2 analysis [Level2].

It would certainly be useful to be able to look at packet payloads as well, though there are both technical and legal difficulties involved. The technical difficulties arise if encrypted communication is being used between the end points, and one has access only to the traffic in between them. Packet payloads may not be available from public providers without appropriate legal procedures to compel them to reveal the data. Commercial enterprises may examine the packet payloads for the traffic involving their employees and their own networks if their policies and guidelines make it clear that there is no presumption of privacy on the part of the users of the network.

Internal Monitors and Log Files

Internal Monitoring refers to observation of the activity of a machine from "inside" that machine, using standard programs intended for that purpose. Some of these programs require the establishment of physical or privileged (root) level control and others only require user level control. Within a private network the tracker can expect cooperation if he works for the network owner. Typically the victim of an attack is willing to cooperate. In fact, the tracker is normally either working for the victim, for the victim's ISP (which may also be a victim) or for law enforcement. For purposes of tracking in the Internet, the tracker can expect little cooperation from most machines, and that includes most of those in the Level 2 causal chain. Machines closer to the victim along the causal chain are more likely to cooperate than others, mainly because they are also likely to belong to the victim. The machines close to the victim tend to be attractive targets for the attacker, since he can often do more damage to the victim by controlling machines close to the victim.

Internal monitors can gather data that could prove useful to understanding the objectives and patterns of behavior of the attackers. Examples of data include:

- Recording the programs that open up network connections or use resources in specific ways.
- Recording the programs that were used by the attacker.
- Recording accesses to privileged documents and other assets of an organization.
- Recording uses of email, chat, and other services.
- Recording documents generated by potential attackers.
- Tracking shell command history, file transfer history, mail history, and other relevant histories.

A "*honeypot*" is considered to be a very special case of a machine with a highly non-standardized, privileged, internal monitor. A honeypot is instrumented in special ways to gather a great deal of fairly low-level data such as calls to specific system functions, etc. A honeypot is set up specifically to gather data without being detected by attackers. Most honeypots are anonymous machines, and are, therefore, best used to detect attacks that are directed at arbitrary machines.

A Log is data recorded (quite possibly by an internal monitor) specifically for the purpose of recovering information about *past activity*. The advantage of a log over observation of current activity is that it is still available even after the activity ends. The disadvantage is that normally much less is recorded than could be observed when the activity is actually in progress. What logs are available varies with each machine. The data specifically produced via logs is not any different from that produced by internal monitors. Internal monitors support real-time tracking activity, whereas logs are persistent, however, and, therefore provide access to past data in a way that internal monitors cannot.

III.2.3 A-Priori Modeling of Threats and Vulnerabilities

In the discussion about the different kinds of data upon which different Level 3 attribution rely one cannot omit the general security condition of the victim site and its understanding (or lack thereof) of that condition. The results of the analysis of the site's vulnerabilities and threats are represented in a model. Such a model is a very different kind of data than the other categories of measurables as discussed above. Nevertheless, such a model could prove to be crucial for every potential victim site to perform attribution in a deliberate fashion. Moreover, since this model of threats and vulnerabilities is one of the few pieces of data that can be gathered in advance of any incidents and without any cooperation from anywhere, it would seem to behoove every enterprise to have such data available as part of its standard security analysis and incident response. Analysis of the model after an incident provides an indication of the attacker's goals and capabilities – important to Level 3 attribution.

When an incident occurs it is because the attacker has found some way around whatever defenses the victim may have erected. In every successful attack activity, the attacker has exploited either a known vulnerability or unknown vulnerability. The known vulnerabilities should be reflected in the model, and the model will reveal the ways in which those vulnerabilities could be exploited. If the attacker has exploited an unknown vulnerability that is not represented in the model, that fact also provides information about the attacker's goals and capabilities. This is relevant to attribution.

There are different techniques that one might use to model the vulnerabilities and threats to one's infrastructure, which as discussed in Sections III.3.1 and III.3.3.4.

III.3 Description of Related Work

This section presents a review of related work using the framework just described.

III.3.1 Measurables for Level 3 Attribution

Natural Language Documents

Determining the authorship of documents has a long history. For example, there has long been controversy in literary circles about whether Shakespeare really wrote all the plays he is credited with writing. In America, there are historical disputes as to the author(s) of the Federalist Papers, a collection of 77 essays written in the 18th century [Fung]. Human beings tend to argue about authorship based upon analysis of the content and style of the documents, and by comparing the disputed documents with documents that are widely acknowledged or known to be written by specific authors.

In a forensics context, investigators often find documents that have presumably been written by authors whom the investigators would like to identify because they are connected to the investigation. Examples of such documents include ransom notes or anonymous letters that law enforcement officials encounter in the course of an investigation. In the present problem, it is assumed that the natural language documents were obtained through means that are not of our concern – for example, they might have been obtained by seizing computers in an investigation or by examining compromised computers or through an informant. In short, it is assumed that the human tracker has some other way of establishing the antecedents of the documents of interest.

There are some key differences between electronic and physical documents. With electronic documents all one typically has available is the raw text of the document. It may also be possible, in some cases, to tell what tools were used to produce the electronic document (e.g., the document was generated in MS Word and then converted to HTML). Physical notes, on the other hand, provide numerous valuable clues that might be just as crucial as the content of the document. These clues include, for example, the paper on which the note is written, fingerprints and other marks, hand-writing style, type of writing instrument, or the alignment of keys on a typewriter that might have been used, etc. This document will delineate the author characteristics that can be gleaned from raw text using various techniques for analysis of the data in Section III.2.1.

Email and Chat Messages

Email and Chat messages incorporate free text, of course, and are, therefore, subject to the same analysis as all other text artifacts as above. However, Email and Chat messages also provide additional contextual information beyond a free text document that could be relevant to Level 3 attribution. This is because both are based on standard protocols such as Send Mail Transfer Protocol (SMTP) or Internet Relay Chat (IRC). For example, while the actual content of an email could be just as inscrutable (from an authorship perspective) as any other piece of text, the email does incorporate additional information about the sender such as his nominal email address, how the message was routed, the subject field, etc.

The same is true of Chat messages, where the chat message incorporate additional information about the sender's name, IP address, etc., per IRC. In fact, with Chat messages one can assume that the sender of the message (either the author or a program he has constructed) is actually on a computer at the time the message is noticed in the chat channel. While such information is hardly adequate to identify sophisticated attackers, it is nevertheless useful to have the additional context available. The "chat culture" is so evolved that one can argue that Chat has pretty much become a special purpose language unto itself – analyzing it is somewhat different from analyzing pure natural language.

Attack Code and Paraphernalia

When one detects malware on a computer, it is useful to analyze the malware for authorship or modifications. Malware is rarely written from scratch. In order to be efficient, hackers often make modifications to a common script in order to customize it specifically for a given computer or network. Therefore, it can be fruitful to look for similar versions of a piece of malware to learn something about elements that are similar or different. Since such customizations typically involve code or script changes that are visible to the victim, they might provide valuable clues as to the sophistication level of the attacker, his computer language background, his preferred *nom de plume* or "handle", and his programming style. In addition, hackers, eager to gain identification, recognition, and acceptance within their peer communities, tend to leave behind signs that identify specific artifacts as their handiwork [Gibson].

With malware, in some cases, the tracker might have just the executables. While executables do provide information on the source language, compiler, and even modifications (if the tracker has the "original" from which the executable was derived), they are not as useful as having source code or scripts. Although attackers are more interested in minimizing keystrokes than programming in a productive or uniform coding style, scripts or source code (if available) could provide additional stylistic information (e.g., how variables are named), comments, and other visible code features.

Further, in the case where the attacker has to customize general malware to install it in a specific place, he often has to download the source and other tools to build the executable on the victim computer. Such activity often leaves traces in system log files or intrusion logs that can provide additional clues in the investigation. There is a body of literature on code understanding and authoring that is useful for Level 3 attribution.

Keystroke Timings

As long as people have been writing and transmitting data through a variety of devices there has been an understanding of the fact that human beings have unique handwriting styles or unique patterns of interaction with the mechanical devices involved. In World War II, it was well documented that telegraph operators on different U.S. ships could recognize the operator who sent a telegraph message from the "fist of the sender" – namely, the unique telegraphic pattern of the author [Telegraph]. Many a whodunit plot is based upon the detective using the unique characteristics of typewriters (the alignment of keys on the keyboard, for example) to deduce the device upon which a note or letter was written, and thereby track down the perpetrator of a crime. The modern-day equivalent of these works is analysis of keystrokes at a computer keyboard.

There is considerable research work that seems to reinforce the uniqueness of "keystroke rhythm" of different human beings at their computer keyboards [Bergadamo, Bleha, Brown, Joyce, Kitchens, Lustig, Monrose, Song]. The dynamics of a person typing at a keyboard is broken down into many different lower level measurements such as:

- *Keystroke duration*: how long is a particular key pressed down? The duration often depends on the specific key, and perhaps the surrounding keystrokes.
- *Keystroke latency*: how long between particular keystrokes (note this varies based on the actual keys involved)
- More rarely, one also sees measurements of the degree of pressure applied on each keystroke, and the placement of fingers on the keyboard

The hypothesis behind gathering the above data is that, for recurring, regularly typed strings that are well known to the human being (such as user names or passwords), keystroke patterns can be consistent enough to be used as a form of signature. Some researchers extend the hypothesis to unfamiliar strings of arbitrary length and complexity. Thus, keystroke patterns can form the basis for additional authentication, essentially providing further assurance that the person typing in the user's name and password actually is the person to whom that user name and password belongs.

Most researchers do not seem to view the keyboard device itself as a key variable in keystroke dynamics. It would seem to be interesting to know, for example, that two patterns of keystroke timings are from the same user, but from different keyboards. Or, perhaps, one can deduce that the keystroke timings match those of a Macintosh user on a PC keyboard, and so on. However, research has not been found that factors in the variations caused by keyboards.

Historically, keystroke analysis has been used far more in an authentication (rather than an attribution) context. Although some of the analysis is similar in both cases, authentication is an easier problem than attribution because one has an established set of baseline signatures to use for comparison. Further, with authentication one can readily imagine that custom software and/or hardware could be installed for authentication at key access points in order to unobtrusively gather the necessary keystroke data for each user attempting to perform the authorized access. Most authentication techniques simply assume that such data can be gathered.

In contrast to authentication, when the problem is attribution, there are major differences in how these assumptions may play themselves out:

a) **Lack of Baselines:** With attribution, here is no closed list of "authorized" signatures. Unlike fingerprints, where a large common database of fingerprints is maintained by law enforcement, no such database has been collected for keystroke signatures. If a particular keystroke pattern does *not* match the existing set of signatures, one can only draw somewhat weaker, nevertheless useful, conclusions – for example, it might be interesting to know that a particular keystroke pattern of interest in an incident had also been observed previously on July 14, 1984.

b) **Reliability of Keystroke Data:** Moreover, with attribution, there are more serious challenges in getting keystroke data about the attacker. For example, the attacker is not going to install any special software or hardware to help one gather the relevant data. With attribution, even in the best cases, one is likely to have access to no more

than network traffic at locations quite removed from the attacker. In such cases, depending on the application (e.g., whether it is encrypted or not or sends individual characters in packets), one might be able to recover enough raw keystroke data to perform meaningful analysis.

Attack Models

One popular form of an attack model is based on attack trees [Schneier]. Attack trees are AND/OR graphs trees whose roots correspond to the assets that might be compromised. The tree itself represents how a potential attacker might end up achieving the objective represented by the root. The tree contains both AND nodes and OR nodes. An AND node represents an objective that can be fulfilled only by fulfilling all the objectives represented by each of its children. An OR node represents an objective that can be achieved by fulfilling any one of the objectives represented by its children. The leaves of the tree represent raw capabilities that the attacker might possess. Attack trees allow security professionals to represent essentially all of the known ways to compromise their computing resources. When an attack is under way, the attack tree could be a valuable instrument in showing how the attacker might have achieved his objective, what capabilities are implicit in the path he used to mount the attack, his overall goals, and other valuable information for attribution. An example of how to use attack trees appears in [Moore].

[Liu] describes a game-theoretic model to model attacker intentions and behavior. The authors advocate the a-priori creation of a model that provides guidance on the potential attacker's intent, objectives, and strategies. The techniques they propose provide the basis to analyze (and even predict) the mechanisms that attackers are likely to use in the context of a given system that they would like to compromise. Comparison of different attacker strategies can be supported to assess the incentives for the attacker to choose one over the other.

III.3.2 Techniques For Level 3 Attribution Using Single Measurable

This section covers the techniques that are employed along with the different data described to draw inferences about the human being involved in a particular situation.

III.3.2.1 Document Analysis

There is a considerable body of work on analyzing the authorship of a document, which is related to the Level 3 attribution problem. In general, these methods examine natural language documents to deduce attributes about the author in various discrete areas as summarized below:

a) Knowledge of the Author:
   - Language used in the document
   - Errors in spelling and grammar, and degree of correlation with native authors or other groups
   - Idioms and cultural references
   - Level of education
   - Specific demonstrated areas of expertise
   - Holes in knowledge

b) Attitudes, Beliefs, and Goals of the Author:
   - Political, religious, ideological content
   - Overall Intent of the author

c) Idiosyncrasies in Language Use
   - Use of specific words, catch phrases, idioms
   - Punctuation
   - Humor

d) Personal Characteristics
   - Gender
   - Nationality, race or other cultural indications
   - Knowledge of specific argot

Computer-aided determination of authorship works reasonably well with the simpler problem of establishing who, from a universe of known authors, is most likely to have written a particular document. For example, computers have been used to analyze The Federalist Papers for authorship to determine which of them was authored by which combination of the known collaborators Hamilton, Madison, and Jay [Fung]. The problem of Level 3 attribution is somewhat more complicated because there is no baseline to compare it to.

Traditional Artificial Intelligence (AI) techniques have been developed to understand natural language. While these methods span a variety of different techniques, most include an explicit representation of the grammar (syntax) of the language in question, and the meaning or semantics of the language in question. Unlike most computer languages, which are based largely on context-free grammars, natural languages are known to require powerful context-sensitive or phrase-structured grammars. Parsing natural language is known to be a very hard problem, and representing the semantics of the language is even harder [Rieger].

These approaches were originally intended to get programs to "understand" natural language in the same sense that humans understand natural language. However one defines understanding of language, it is clear that the results remain quite far from having programs able to do what humans do with natural language. At this time, natural language understanding has had limited success in areas like allowing users to use simple natural language in interfacing with different kinds of software including search engines, help systems, etc. However, true natural language understanding has proved to be computationally intractable for anything that resembles non-trivial text [Damereau].

For some aspects of Level 3 attribution, however, one does not need to really understand the text being analyzed. It is adequate to establish the similarity between the text being analyzed and text that is written by a known author. Several researchers conduct what one might call "non-linguistic" analysis based purely on statistics about the document. These statistics constitute something considerably less than whatever one might term "understanding" of the document, but there is enough capture of the style and characteristics of the author to make the techniques useful in attribution. These techniques have become more prevalent (and somewhat more successful) than natural language understanding programs because they are computationally tractable and scale better for larger documents.

The information theorist Shannon [Shannon] early saw the possibility of using statistical estimation techniques to solve language-related problems such as prediction of the next word given a particular sequence of words. This area has evolved into a vigorous area of research with several techniques available. The good thing about statistically based techniques is that they extend easily to languages other than English. In fact, the techniques discussed in this section have been used with Arabic [Darwish] and Asian languages [Suzuki].

The *Bag of Words* techniques focus upon viewing the text as simply a collection of words, each of which occurs a certain number of times. Certain common words could be removed from the sample if one wants a more refined *bag of words* measure. Text can be categorized using various statistical schemes upon the *bag of words* data, including naïve Bayes classification or Hidden Markov Models (HMM) and other methods. HMM is described in Section III.3.3.2.

Naïve Bayesian classification can be used with various kinds of data, including bag of words. The reason the method is considered "naïve" is because the words are considered independent of one another – clearly not true. Using classical Bayesian theory, a naïve Bayesian classifier would try to find a class C that has the highest probability to explain the existing distribution in the bag of words. Despite its somewhat dismissive name, naïve Bayesian classifiers are documented to work well in many cases [Heckerman].

Chi Square is another simple statistical technique that has been used to determine whether a given document was authored by a particular person, given samples of previous documents written by that person. The technique involves selecting a set of words at random, and comparing their expected and observed frequencies from the sample and the text being analyzed respectively. The Chi Square formula is then computed using a 2 X 2 "contingency" matrix that models the expected and actual values. If this value is greater than 3.84, then there is less than 5% probability that differences between the observed and the actual data is due strictly to chance, and the hypothesis of authorship can be rejected. The Chi Square technique has shortcomings because it requires that words be drawn at random from an independent identically distributed ("i.i.d.") distribution. This is almost never the case in documents because different words are not identical in their distributions. Variations of Chi Square that focus on the "n" most frequent words (rather than words drawn at random) seem to be a better measure.

Another popular technique for language analysis is using the idea of "n-grams". With n-gram analysis, the input string is broken up into word sequences of a particular length. Sequences of length 1 are called "unigrams", sequences of length 2 are called bigrams, and sequences of length 3 are called "trigrams". This could go on for arbitrarily large values of n. The set of elements of any particular length for the initial input text is termed the "training set". For smaller values of n, one gets many more elements in the training set, which aids reliability at the cost of context discrimination. Larger values of n yield fewer elements in the training set, which aids context discrimination at the cost of reliability.

For any given "n", the n-gram analysis yields a probability distribution function over the set of unique n-grams. These probabilities are used with the Bayes chain rule to predict the likely nth word given the preceding n-1 words in a sequence. N-gram analysis has been used with a variety of statistical algorithms [Soboroff] and has experienced varying success.

X-grams are an extension of N-grams presented in [Bonafonte] that obviates the need for the language modeler to select the value of "n". Instead, X-grams enable the algorithm to select the value of "n" using the information within the training set itself. This research shows that X-grams have certain complexity advantages over N-grams when it comes to using them for recognition.

[Diederich] shows the utility of Support Vector Machines (SVMs) in analyzing text documents to extract authorship characteristics. SVMs have been attractive in the text categorization domain because they are capable of mining inputs for hundreds of features, which is considerably better than neural networks and decision trees. The work looks for stylistic elements of a document that are assumed to be beyond the conscious control of the author (even if he wanted to obscure his identity by altering his style). SVMs are described further in Section III.3.3.3.

Overall Assessment of Document Analysis techniques:

- Linguistic techniques still remain computationally intractable, but have the best prospects for accuracy of results.
- Reliability of statistical methods varies greatly with application. Authorship attribution is not perfect, but seems to work reasonably well if there are samples of the author's other documents available for analysis.
- How hard will it be for someone who knows how the methods work to deceive the methods, and throw the analysis off? It would appear relatively easy for human beings who know the analysis methods to deceive the algorithms.
- For Level 3 attribution, given that the problem is hard, it may be useful to have such analysis available – perhaps using multiple methods to offset the unreliability of individual ones.

III.3.2.2 Analysis of Email and Chat

[de Vel 1, de Vel 2], in presenting a data mining framework for email, points out some of the key differences between email analysis and more general language analysis. For example, email analysis has access to additional information about the email messages such as the header information (which has the nominal sender address), time stamps, routing information, etc. However, because of increasing ability of attackers to spoof such data, it is important to treat them as additional information to be verified.

Email and Chat authorship differs from text categorization in general in a couple of ways. Prompted by the desire to cut down the number of keystrokes, people tend to use their own stylized contractions for common phrases, shortcuts, and signatures. These aspects provide additional clues about authorship that is not present in regular documents. Such observations are particularly true of Chat, where one can argue a whole new language has been invented.

[de Vel 1] shows techniques that can be used in authorship categorization, which is the problem of determining if a given email is written by a specific person given other samples of email that the person is known to have written. Unfortunately, email messages tend to be much shorter than normal text documents, and that tends to limit the full expression of personal style. The technique used here features SVMs.

[Argamon] presents email analysis based on computational stylistics. Their research focuses upon the document style rather than content. The algorithm they use, which is a variation of the Exponentiated Gradient (EG) learning algorithm. This is a statistically based algorithm that is claimed to outperform Bayesian and other analysis for the email domain. The approach has been validated using newsgroup exchanges (for privacy reasons email was not used).

[Corney] presents techniques that demonstrate how one might analyze email to deduce the gender of the author. Using Support Vector Machines (SVMs) as the underlying computational vehicle, the research focuses on gender-based differences in communication as it relates to email in particular. They focus on "emoticons" – various stylized ways of expressing low-intensity emotions, spelling of certain words, and other key differences between men and women in their email communications.

III.3.2.3 Keystroke Analysis

The idea of using keystroke dynamics to perform computer user authentication seems to date to the early 1980s, when a RAND corporation study [Gaines] documented that the keyboard dynamics were individualized and repeatable, at least to the degree needed to discriminate between the typing patterns of each member within a small population. The analysis used was keystroke digraphs – keystroke latencies between successive keys. The digraphs showed considerable variation for different pairs of keys, but were the same for the same pairs of keys.

Other experiments [Umphress, Legett] extend the digraph latency analysis to incorporate speed of typing, separation of left from right hand, and other variables. They also corrected the flaws in the RAND study resulting from the very small population size. These studies also seemed to indicate that using keystroke dynamics for identification is viable. Two patents [Garcia, Young] appear for the use of keystrokes for personal identification in 1986 and 1989 respectively.

The above keystroke analysis requires the typing of unfamiliar text of at least 1,000 words to compute the baseline digraph latency matrix. Some researchers [Joyce] believed that the patterns for unfamiliar text would not be a good baseline and that the error rate could be unacceptably high. [Joyce] uses 4 familiar strings typed in by the user to get a more consistent set of digraph latencies (e.g., first name, last name, user name, password). While earlier techniques looked simply at statistical deviations of the actual signature from the baseline, [Joyce] allows for comparison of the shapes of the signature – which is a way of taking into account the surrounding context (i.e., keys preceding and following each digraph) for each digraph as potentially affecting the digraph latency.

The reliability of the [Joyce] work was further improved upon by [Fabian] in several ways. First, different training sets recorded at different times are used in the scheme. Second, more comprehensive classifiers were used in the algorithm such as Euclidean Distance Measure, Non-Weighted Probability, and Weighted Probability between the two vectors representing the baseline and signature being identified.

A more powerful Bayes classifer is adapted for use in keystroke analysis of user names and passwords in [Bleha-1]. The rate at which impostors are allowed in by this algorithm was 2.0% – which is unacceptably high for authentication, but may not be as much of a problem for attribution.

In contrast to the above works, which use statistical techniques, others use a machine learning approach to the keystroke analysis problem. This is an approach that features continuous monitoring and improvement of the performance of the algorithms over time as they become "trained" to do better.

A couple of examples of using neural networks in keyboard analysis include [Brown, Lammers]. [Lammers] describes a project where neural networks are used with keystrokes to determine identity. The argument made in this research is that rather than using a sequential approach, neural networks can explore multiple hypotheses simultaneously. [Brown] discusses the use of neural networks as well, and this work is the subject of a patent on "apparatus to verify a computer user's identification."

Another, more recent, machine learning approach [Song] uses continuous monitoring of keystrokes so that there is continuous machine learning of the user's keyboard dynamics. Earlier analysis is based on latency between keystroke events. One distinction that is found in the [Song] research is the use of lower level Key Press and Key Release events. This tactic has a couple of different implications: one can distinguish between "RP latency" – the latency between a key being released and the next key getting pressed – and the "PR latency" – the latency between a key being pressed and released. In addition, this method uses "bigrams", which is the grouping of two continuous keystrokes; and "trigrams", which is the grouping of three continuous keystrokes; and "wordgrams", the grouping of continuous keystrokes that together form a word. A Markov model is proposed as the basis for analyzing the data, in addition simple statistical measures.

Overall Assessment of Keystroke Analysis:

While some techniques have worked better than others, keystroke analysis, as a whole, has just not been accurate enough to do authentication [Bergadamo, Song]. This is the reason why the technique has not taken hold in the field of biometric identification, despite considerable commercial interest in an era of heightened awareness about security. Allowing impostors in even at a rate of one in 10000 (which is the best documented performance by a keystroke analysis method [Bergadamo]) is completely unacceptable for authentication. However, it might still be adequate for attribution, which is our purpose in this project.

One of the major problems is the underlying assumption of all keystroke analysis: that keystroke dynamics are stable for each human being. The evidence for this belief is not exactly overwhelming. [Bergadamo] documents the poor results of existing methods, and posits that they fare poorly because the monitored parameters depend too much upon the mood and psychological or physiological state of the human being. [Bergadamo] compensates for such effects assuming "homogeneous" deviation, namely, that the deviations from the norm for a given person at a given time, if any, are ALL in the same direction. Even with such assumptions, the error rates are unacceptably high for authentication.

The keystroke analysis methods, while being inadequate for authentication, might still provide good information for attribution. Attribution is a much harder problem to solve than authentication, justifying a more aggressive approach. Moreover, in Level 3 attribution, it is assumed there is always a human in the loop. Among other things, the human can be used to offset false positives and false negatives. Another point is that keystroke dynamics might be

much more effective in distinguishing between the members of a small population, for example. Supporting such "sub-problems" seems like a better fit for the kind of overall reliability that keystroke analysis provides. An important question to examine would be whether users can "spoof" their keyboard patterns to throw off such analysis.

III.3.2.4 Attack Code Analysis and Attack Code Eggs

Analysis of software has been motivated by reasons such as figuring out the inadequacies of software code, or reasoning about the similarity between pieces of code to determine plagiarism [Ribler], or establishing authorship in forensics context [Weeber]. Code eggs are hidden, embedded programs that are occasionally included inside other programs sometimes for fun, but also to do harm. It could be important to detect code eggs and their authors where applicable as well.

[Weeber] examines the potential to identify the author of various kinds of malware from fragments of code that are left behind. While conceding that the author of code who knows that the code will be analyzed for authorship will likely be able to throw off the algorithms, this research nevertheless provided an approach to analyze the authorship characteristics. If only executables were available, one analyses data structures and algorithms used, compiler and system information, programming skill and knowledge, choice of system calls, and errors. Source code provides a far richer base for analysis of authorship of the program, of course. From the source, one knows the language being used, formatting style, comments, variable names, spelling and grammar, and use of other language features.

Commenting on the general potential of "software forensics", namely identifying the authorship of programs, [Sallis] points out the software metrics that correlate closely with authorship. These metrics include layout metrics, style metrics, and structure metrics. Using these metrics, about 78% of the authors were identified correctly. This work suggests that considerably more work needs to be done to individualize the programmer profile to incorporate elements of style similar to natural language processing. The recommendation is to use a wide variety of stylistic metrics to get authorship data from many different perspectives.

[Krsul] argues that program analysis is harder than language analysis because of the collaborative way in which most software is built. When there is a melding of multiple styles of development with a larger team, the problem becomes much harder. On the other hand, natural language is also a very difficult problem to resolve.

[Krinke] presents a way to compare pieces of code for similarity using the technique of finding similar subgraphs between 2 different attributed directed graphs. The technique uses a full representation of the syntax and semantics of the language, program dependency graphs, and data flow. The technique is demonstrated to be successful through experimentation.

[Ducasse] discusses a language-independent approach to analyzing duplicated portions of two different programs using simple string pattern matching, textual reports of differences and similarities, and scatter plot visualization of the comparison. This approach intentionally eliminates the need for parsers because it deals with a variety of languages and dialects. It seems to be particularly successful in handling long programs. It is an advanced version of the UNIX *diff* utility, relying on a more sophisticated string comparison algorithm.

[Ribler] describes a system for visualization of programs that helps detect when the students who wrote them might have plagiarized the programs or portions thereof. The technique combines N-grams with pattern-matching and visualization techniques. The algorithm tries to account for renamed variables and other obvious tactics to hide the plagiarism. The approach is embodied in a tool called Chitra, and has been validated by instructors in their programming courses.

[Kontogianos] discusses an approach to detect patterns in software. The technique was created originally to identify software "clones" that are the source of redundancy and bugs in software. Each program is transformed into an abstract syntax tree, and patterns are specified in terms of this data structure by focusing on specific parameters that identify data flow and control flow. The approach has been shown to scale well from a performance standpoint, but precision drops dramatically for larger examples.

[Gray] proposes a framework for analysis of malware that can provide useful clues about the author. He proposes gathering metrics about the time when the code was written, programming language, compiler, tools used, formatting of code, environment used, macros, comments, variable naming style, spelling, grammar, preference for specific language features, code size, execution errors, reuse of previous code, selection of data structures, selection of algorithms, sophistication of programming knowledge, knowledge of system and library calls, and coding errors. The author's group has produced a piece of software called IDENTIFIED that embodies this proposed framework.

Overall Assessment of Attack Code Analysis:

The actual, proven record of the techniques discussed in these research works here is hard to verify. Tools to compare programs to other programs for similarity seem to be rather more successful than tools that try to glean author characteristics. These tools have largely been used to verify plagiarism, and largely within the University context. This, arguably, does have some relationship to the problem of understanding the characteristics of the author of a program by analyzing the program.

These tools appear much more likely to succeed if the problem becomes one of checking whether a given program is likely to have been written by a programmer given several other sample programs authored by that programmer. Such a problem could indeed be relevant to Level 3 attribution.

III.3.3 Integrating Multiple Characteristics

Different techniques operating on different online data provide us valuable information about the characteristics of the human beings involved in specific incidents. How can the information from different techniques be integrated to form a single picture? Moreover, not all the data gathered and/or analyzed in law enforcement contexts is online. How can one take ALL known information (from the cyber universe and the real one) and integrate everything into a single composite picture? Many different statistical techniques exist. Note that some of these techniques are used to analyze single measurables as well.

III.3.3.1 Bayesian Networks

A Bayesian network is a graph that encodes the probabilistic relationship between many different variables of interest. While formal Bayesian theory is based upon notions of conditional probability and independence, in practice, a Bayesian network is often viewed as representing causal relationships in a problem domain. This allows the Bayesian network to simultaneously provide a representation for prior knowledge and also provide a way to incrementally incorporate new data (i.e., learning). One of the advantages of the Bayesian network is that it is able to incorporate situations where the data are incomplete, which is the norm in our problem domain.

A wealth of statistical techniques is available for Bayesian networks that could prove to be very useful in Level 3 attribution:

- *Bayesian Inferencing*: Bayesian inference can be used to compute the probability of any outcome of interest, given specific observations. There are well over half a dozen different inferencing schemes that use Bayesian networks. A full treatment of all these techniques would be beyond the scope of this paper. For our purposes here, it is sufficient to note that the capability exists.
- *Handling Incomplete Data*: Several well-evolved techniques exist to use Bayesian networks in the context of incomplete data. There are two cases to consider. In the easier case, the absence of data is independent of the state. In the harder case, the absence of data is dependent on the state. A variety of statistical techniques of different complexities exist to handle both cases.

Bayesian networks are supported by a variety of different mathematical and statistical software platforms. They could be one way in which one could integrate multiple variables that represent different characteristics with differing levels of certainty.

III.3.3.2 Hidden Markov Models

While the theoretical foundations of HMM have been known for well over 3 decades, it only in recent years that computer scientists have begun to understand the full range of applications in which HMM could be applied [Rabiner]. There has been particular interest in using HMMs in applications such as text categorization, keystroke analysis, and so on. These problems are of obvious importance in Level 3 attribution. HMMs also provide a basis to unify analysis from multiple perspectives. In this section an overview of HMMs is provided.

Discrete Markov processes enable the description of systems in terms of state changes at regularly spaced time intervals. After each time interval, the system undergoes a change of state according to a set of probabilities associated with the state. In general, the probability of a transition to a particular state depends on all the predecessor states. In most cases, however, researchers focus on a discrete, first-order Markov chain where the probability of a transition to a particular state depends only on that state and its immediate predecessor. This is an observable Markov model that turns out to be far too restrictive to apply to many problems of interest because one would need to be able to observe every event.

In a “hidden” Markov model, the observation is a probabilistic function of the state. The resulting model essentially provides for two processes, one being embedded in the other. The first process is not observable, and is thus hidden, but it can be observed through the second

process whose events produce the sequence of observations. This provides a better paradigm for many real-world problems where the observation of important phenomena can only be effected through another, more noisy process. Thus, a HMM can be characterized by: the set of states in a system, the set of distinct observable symbols, and the state transition probability distribution.

HMMs can be used to answer different questions. In the context of Level 3 attribution, the main question of importance has the following flavor: given a sequence of observations and a particular model, how well does the model match the observations? From a HMM perspective, the Level 3 attribution problem may be viewed as one of finding a model that has the greatest probability of being the correct one, given the set of observations.

A complete treatment of HMMs is beyond the scope of this document. However, there are several online tutorials and learning materials [Rabiner, Moore]. Matlab and other platforms for statistical analysis now provide support for HMM analysis.

III.3.3.3 Support Vector Machines (SVMs)

There has been considerable interest in SVMs in the last few years. While a complete treatment of SVMs is beyond the scope of this paper, a brief overview is provided, sufficient to assess their importance to the Level 3 attribution problem. Tutorial materials may be found at [SVM].

SVMs are a special case of a more general class of machine learning algorithms that are called "kernel methods". Kernel methods and SVMs are good at detecting and manipulating complex patterns in large data streams. Techniques used include clustering, classifying, cleaning, and ranking the data. SVMs have been particularly effective at solving computational problems such as representing complex patterns efficiently, while also dealing with statistical problems such as avoiding what is called "overfitting", which occurs when spurious or unstable data are not detected and excluded. SVMs have been used both to analyze single measurables (e.g., biometric applications such as recognition of hand-writing) and to correlate multiple perspectives.

The class of kernel methods defines implicitly the set of patterns of interest by choosing concepts of "similarity" between data. In so doing, one is choosing the features of importance in the problem. Kernel methods use the concept of inner products between data items within a particular feature space. These methods can be quite complex, but once they are defined within a kernel, there is no need to specify what features of the data are of importance. The kernel learning algorithm has both a general machine learning component and a problem-dependent component. This feature combines the reusability of formal analysis with the requisite non-standardizability needed to fit the analysis to a specific problem situation.

SVMs solve some of the problems associated with linear learning machines that use traditional regression analysis to estimate the best relationship possible between the different data. In particular, linear regression methods tend to break down if the data are noisy or if the data are not linearly separable. Before SVMs, the solution was usually to move to a non-linear classifier such as a neural network. The problem with neural networks is that they require many parameters and must be trained using the appropriate heuristics in a supervised fashion. With SVMs, the solution is to map the data into a richer feature space that includes non-linear aspects,

and then use a linear classifier in each dimension. This can become computationally complex, but SVMs provide a way to keep things manageable in most cases.

III.3.3.4 Self Organizing Feature Maps

Kohonen's self-organizing feature maps (SOMs) [Kohonen] are a data visualization technique that can reduce the dimensions of data through the use of self-organizing neural networks. Since humans simply cannot visualize high dimensional vector data, SOMs can be very useful in showing humans potential similarities and differences between different data. SOMs are able to produce a map of no more than 1 or 2 dimensions in most cases. SOMs help human beings in two major ways. First, they reduce the dimensions of data; and, second, they focus on displaying similarities in a way that human beings will notice them. One of the most interesting aspects of SOMs is that they "learn" to classify data without supervision.

There are two important aspects that are crucial for a SOM to work:

- *Data*: Raw data corresponding to all the different dimensions of information are required for the SOM to function.
- *Weights*: One also needs a weight vector – an assignment of importance to each piece of data.

The SOM then uses a self-organizing, neural network based algorithm to assist in reducing the complex dimensional data into something that can be visualized on a 2-dimensional screen without losing sight of parts that are similar.

SOMs are effective in classifying the data. They are particularly intuitive for humans to understand patterns of similarity in data. Other formalisms do not offer the visual display of similarities as SOMs do. Further, because of its self-organizing nature, the structure adjusts itself to do even better on a continuous basis.

The biggest practical obstacle to using SOMs is that they require all the data in all dimensions in order to function. For many problems (including the Level 3 attribution problem), this is a difficult requirement to satisfy because the data will always be incomplete for a variety of reasons such as lack of available cooperation, for example. A map cannot be generated without all the data for each dimension being available. A technical problem with SOMs is that different SOMs might find very different similarities among the sample vectors. This can have misleading visual impact. To reduce the likelihood of this kind of problem one needs to increase the number of maps considerably.

III.3.3.5 Analysis of Attack and Attacker Characteristics

One of the problems in conducting Level 3 attribution is that there is the need for a human tracker in the loop. Further, there are huge gaps in the kind of coverage provided by all the different kinds of techniques and tools that have been discussed. It is, therefore, imperative that the tracker's work of attribution be situated in some overall context of information that is trying to reconstruct a plausible way in which the activity was perpetrated. A variety of correlation and analysis tools may be useful here, but they need to be guided by an experienced human tracker. Hence, this final piece of "integrating" multiple perspectives that must be performed by the tracker himself.

The investigation of an incident should be conducted with knowledge of attack models (discussed in Section III.3.1) that exist for the organization. This information must be correlated with all the other data that is available about the activity that is being attributed. This analysis can reveal interesting aspects about the attack and the attacker. In practice, no attack model will be complete no matter how conscientious and competent the people who construct it. However, the attack model can nevertheless prove to be of value in forensics analysis in highlighting processes the attacker might have used.

[Ning] discusses a process to use intrusion logs and alerts to learn about the attacker's strategies. The approach represents relationships between intrusion alerts in a graphical form. Specific sub-graphs that are seen repeatedly are highlighted as strategies used by attackers. When these strategies are later encountered, they provide information about what the attacker is after much more rapidly than trying to correlate the individual alerts each time.

The attack trees of [Schneier] and the game-theoretic models of [Liu] can also be used as the foundation for further forensics. By comparing the actual evidence with the kind of attack trajectory that the models would predict the tracker could uncover the methods used by the attacker.

The tool called SilentRunner [SilentRunner] provides a common platform to perform different kinds of automated analysis to aid the tracker. As a commercial product, it is now hard to get detailed technical information about this software. However, it seems to unify several important algorithms discussed here in the context of authorship with an ability to handle large intrusion logs and traffic monitoring. SilentRunner seems to be able to perform many different tasks to aid the human tracker. For example, given a large volume of data that documents communications between different human beings (such as call records), SilentRunner can reliably find patterns that indicate collaboration, hierarchy, which person is in charge, and other such "organizational" characteristics. As a result, this software has been successful in aiding human beings in making sense out of a huge volume of data, while allowing the human being to determine the importance and implication of the analysis in a larger investigative context.

The human tracker must recreate a picture of the attacker's objectives, capabilities, and strategy to perpetrate the observed activity. The techniques discussed herein can provide the foundation for the tracker to construct such a picture.

III.4 What Can and Cannot be Done with Current Technology

This report first summarizes what one can achieve with individual methods that use particular measurables, and then comment on the overall shortcomings.

Measurable & Method	Characteristics Uncovered	Effectiveness
Document Analysis – Natural language methods	Attacker goals, style, education, native language, knowledge; comparison to prior writings	Computationally intractable; but potentially more accurate
Document Analysis – statistical methods	Attacker goals, style, education, native language, knowledge; comparison to prior writings	Computationally tractable; probabilistic answer is provided; Attacker might be able to deceive the analysis
Keystroke Timing	Comparison to prior profiles; left-handed or right-handed;	Computationally tractable, but results are unreliable because the attacker can mislead the analysis
Email Authorship	Similar to natural language; gender;	Potentially useful – similar problems as document analysis
Attack Code Analysis	Attacker's sophistication level; tools used; knowledge; capabilities and resources;	Potentially effective; also no need for cooperation with anyone else;
Attack Models	Enumerate potential paths for attacker to take to perpetrate activity	Starting point for Level 3 attribution process

Table 3.1.

When it comes to Level 3 attribution, the problem is not that there are no tools. There are *individual* tools (as indicated above) that can prove useful in specific situations. The real problem is that individual tools is *all* there is at this time. How is information from one tool to be integrated with or correlated to deductions made by another? It is clear that the gaps between the tools must be filled by the cognitive power of a human tracker who understands the "big picture" and is able to orchestrate all the tools together so that the support his goals. There is a big technology gap in the area of forensic tools being able to integrate smoothly with many different kinds of repositories and sources of critical information that need to be correlated.

Another major problem for the entire attribution area is the assumption of data availability and cooperation. Without data, no analysis can succeed. This is not a shortcoming of the methods, but it is more likely to bring an investigation to a halt than the ineffectiveness of a particular kind of analysis. This issue of cooperation is analyzed more carefully in Section III.5.

III.5 Impact of Cooperation

This section examines the impact of cooperation on Level 3 methods from different perspectives.

Table 3.2. **Impact of Cooperation on Individual Methods**

Method	Impact of Non-Cooperation
Document Analysis	• If the document was obtained inside one's own network, there is no impact • If the document was obtained through cooperation with someone else, then non-cooperation means no document is available
Email & Chat Analysis	• Email: if the email was gathered within a network belonging to a private enterprise, the email can be analyzed so long as the organization's policies make it clear that there is no presumption of privacy • Email: within a public service provider, one is not allowed to analyze any specific email contents • Email: law enforcement can compel information to be revealed by any organization if there is probable cause • Email outside one's own network can be very difficult or impossible to get access to • Chat: for a particular channel, chat can messages can be observed by using a program called a "robot" that pretends to be a human participant listening in. Chat operators don't like robots, and usually have policies to get rid of them
Keystroke Analysis	• Keystroke information can be recovered by examining packets at the victim site so long as the communication between the end-points is not encrypted • Keystroke information from packets can retain useful latency and other properties even far away from the attacker – making them quite attractive in this regard • If the communication is encrypted, then the information about the actual keys used is not recoverable. This makes the analysis much less useful, although not entirely useless!
Attack Code	• This measurable is detected (either in binary or source) at the victim site and is not impacted by non-cooperation in any way

Different methods work on different measurables. The question then becomes whether it is likely that the measurable necessary for a method to work will be available regardless of whether there is cooperation or not. It is assumed that the victim's own infrastructure is willing (and able) to cooperate with the tracker. Therefore, a particular method could be adversely impacted by non-cooperation if it relies on data that is not available within the victim's own infrastructure.

Impact of Cooperation on Data Gathering Methods

This document has discussed different data gathering mechanisms such as network monitoring, running internal monitors on machines, and having access to log files. All these mechanisms are clearly usable within one's own network, and whatever one can find through these mechanisms can be used in attribution.

What happens outside one's own network? A lot depends on the nature of the incident and what the human tracker sees as the process to find the human beings responsible. In many cases, the tracker might have enough data just within the victim's network to perform some (or, in rare cases, even all) of the analysis required. In other cases, however, the tracker might need to do something akin to Level 1 or Level 2 traceback just in order to understand where to focus on next in the tracking investigation, and what kind of cooperation must be sought from the machines identified in the traceback. In this case, cooperation will play a major role. The influence of non-cooperation on Level 1 traceback is examined in [Cs3 Level 1] and the influence of non-cooperation on Level 2 traceback is examined in [Cs3 Level 2]. These issues could also be felt at Level 3 if the tracker tries to identify the causal chain for some activity, depending on the nature of the attack and the kind of process the tracker is using to get closer to the attackers.

Outside one's own infrastructure, if one is using internal monitors or logs, one would need the cooperation of the next machine on the causal chain from which the tracker wants to trace back. If one is using network traffic monitors, on the other hand, one would need the cooperation of a machine in a position to observe and identify the controlling traffic.

Impact of Non-Cooperation on Human Tracker

In the case of Level 3 attribution, there are some cases where the tracker can indeed gain considerable amount of information about the attacker from just the data available at the victim site – attack code, even keystroke dynamics (so long as there is no encryption). In this case, no cooperation is necessary of course. However, once the tracker needs data outside the victim's computing infrastructure, the situation is very similar to that in Level 2 attribution [Cs3 Level 2]. Having roadblocks in acquiring the requisite data from further along the causal chain might well prove to be insurmountable.

In the contemporary Internet, there is no structure for cooperation between trackers. Attackers vastly outnumber the trackers, and often have control of many different machines (zombies) spread out throughout the infrastructure. So, the problem is indeed daunting. Part of the problem is that no organization wants to go public with its security problems. Companies are extremely leery of damage to their reputations and would rather wear their shame in private. There is a trend starting, however, to share more information about attacks for the common good through services like Attack Registry and Intelligence Service (ARIS) [ARIS]. This could prove to be beneficial in the future for trackers in pursuit of attackers. It is expected that cooperation between trackers will be a powerful way to build up Level 3 attribution capability in a network as vast as the Internet, and that this kind of cooperation will be a potentially important topic for research during the course of this project.

REFERENCES

[Argamon] Shlomo Argamon, Marin Saric, & Sterling Stein; *Style Mining of Electronic Messages for Multiple Authorship Discrimination: First Results*; SIGKDD 2003 [August 2003]

[ARIS] Attack Registry and Intelligence Service, http://aris.securityfocus.com

[BAA 03-03-FH] http://www.nbc.gov/pip.cfm

[Baker] Brenda Baker; *On Finding Duplication and Near Duplication in Large Software Systems*; 2nd Working Conference on Reverse Engineering [1995]

[Baran] Paul Baran; *On Distributed Communications: IX Security, Secrecy, and Tamper-Free Considerations*; http://www.rand.org/publications/RM/RM3765/index.html; Rand Memorandum RM-3765-PR [August 1964]

[Baran-Bio] Paul Baran; *Internet Pioneers*; http://www.ibiblio.org/pioneers/baran.html

[Belenky] Andrey Belenky; *IP Traceback with Deterministic Packet Marking DPM*; http://www.library.njit.edu/etd/njit-etd2003-105/njit-etd2003-105.html; Dissertation [2003]

[Bellovin] S. Bellovin; *ICMP Traceback Messages*; http://www.research.att.com/%7Esmb/papers/draft-bellovin-itrace-00.txt; Internet Draft [March 2000]

[Bergadamo] Francesco Bergadamo, Daniele Guneti, & Claudia Picardi; *User Authentication Through Keystroke Dynamics*; ACM Transactions on Information Systems, 5(4) [November 2002]

[Bleha] Saleh Bleha, Charles Slivinsky, & Bassam Hussein; *Computer Access Security Systems Using Keystroke Dynamics*; IEEE Transactions On Pattern Analysis and Machine Intelligence, Vol. 12, No. 12 [December 1990]

[Bonafonte] Antonio Bonafonte & Jose Marino; *Language Modeling Using X-Grams*; International Conference on Spoken Language Processing [1996]

[Brown] ; Marcus Brown & Samuel Joe Rogers; *User Identification Via Keystroke Characteristics of Typed Names Using Neural Networks*; International Journal of Man-Machine Studies, 39(6) [September 1985]

[Buchholz] F. Buchholz and C. Shields; *Providing Process Origin Information to Aid in Network Traceback*; http://www.cs.georgetown.edu/~clay/research/pubs/process_origin.ps; from proceedings of the 2002 USENIX Annual Technical Conference [June 2002]

[Burch] Hal Burch and Bill Cheswick; *Tracing Anonymous Packets to Their Approximate Source*; http://www.usenix.org/publications/library/proceedings/lisa2000/burch/burch_html/; from Usenix LISA [December 2000]

[Cisco] *Characterizing and Tracing Packet Floods Using Cisco Router*; http://www.cisco.com/warp/public/707/22.html

[Cohen-Forgery] Fred Cohen; *Internet Holes – Eliminating IP Address Forgery*; http://www.all.net/journal/netsec/1996-06.html [1996]

[Cohen-Responsibility] Fred Cohen; *Providing for Responsibility in a Global Information Infrastructure*; http://www.all.net/journal/ntb/responsible.html

[Corney] Malcolm Corney, Olivier de Vel, Alison Anderson, & George Mohay*; Gender-Preferential Text Mining of E-Mail Discourse*; 18th Annual Computer Security and Applications Conference [2002]

[CS3 ARDA 1] D. Cohen and K. Narayanaswamy; *Impact of Cooperation on Level 1, 2, and 3 Attack Attribution Techniques*; http://www.cs3-inc.com/pubs/eliminating-source-forgery.pdf

[CS3 ARDA 2] D. Cohen and K. Narayanaswamy; *Algorithms & Techniques Technical Report*; http://www.cs3-inc.com/pubs/eliminating-source-forgery.pdf

[CS3-Forgery] D. Cohen and K. Narayanaswamy; *Changing IP to Eliminate Source Forgery*; http://www.cs3-inc.com/pubs/eliminating-source-forgery.pdf

[Cs3 Level 1] http://www.cs3-inc.com/pubs/cs3-stardeck-briefing.pdf provides a link to the paper entitled *Level 1 Attribution Techniques*

[Cs3 Level 2] http://www.cs3-inc.com/pubs/cs3-stardeck-briefing.pdf provides a link to the paper entitled *Level 2 Attribution Techniques*

[Damerau] Frederick Damerau; *Automatic Parsing for Content Analysis*; CACM, 13(6) [1970]

[Daniels] Thomas E. Daniels; *Reference Models for the Conceralment and Observation of Origin Identity in Store-and-Forward Networks*; http://www.eng.iastate.edu/~daniels/diss.pdf; Purdue University [December, 2002]

[Darwish] Kareem Darwish & Douglas Oard; *Term Selection for Searching Printed Arabic*; SIGIR 2002, Tampere, Finland [August 2002]

[de Lucca] Giuseppe Antonio de Lucca, Massimiliano di Penta & Anna Rita Fassolino; *An Approach to Finding Duplicated Web Pages*; 26th Annual International Computer Software and Applications Conference (COMPSAC) [2002]

[de Vel 1] O. de Vel, A. Anderson, M. Corney, & G. Mohay; *Mining e-Mail Content for Author Identification Forensics*; SIGMOD Record, 30(4) [December 2001]

[de Vel 2] O. de Vel, A. Anderson, M. Corney, & G. Mohay; *Multitopic e-Mail Authorship Attribution Forensics*; ACM Conference on Computer Security – Workshop on Data Mining for Security Applications [November 2001]

[Diederich] Joachim Diederich, Jorg Kinderman, Edda Leopold, & Gerhard Paass; *Authorship Attribution using Support Vector Machines*

[Donoho] D. Donoho, A. G. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford; *Multiscale Stepping-Stone Detection: Detecting Pairs of Jittered Interactive Streams by Exploiting*

Maximum Tolerable Delay; http://www.icir.org/vern/papers/multiscale-stepping-stone.RAID02.pdf; from proceedings at RAID Conference [2002]

[Ducasse] Stephane Ducasse, Matthias Rieger & Serge DeMeyer; *A Language Independent Approach for Detecting Duplicated Code*; Software Composition Group, University of Berne, Switzerland

[Frasconi] Paolo Frasconi, Giovanni Soda, & Alessandro Vullo; *Text Categorization for Multi-Page Documents: A Hybrid, Naïve Bayes Hidden Markov Model Approach*; JCDL 2001, Roanoke, Virginia [June 2001]

[Fung] Glenn Fung; *The Disputed Federalist Papers: Support Vector Machines Feature Selection via Concave Minimization*; TAPIA 2003, Atlanta, Georgia [2003]

[Gibson] DDoS Attacks at Grc.com, http://www.grc.com

[Gray] Andrew Gray, Philip Sallis & Stephen MacDonell; *A Dictionary-Based System for Extracting Source Code Metrics for Software Forensics*; Department of Information Science, University of Otago, Dunedin, New Zealand

[Heckerman] David Heckerman; *A Tutorial on Learning with Bayesian Networks*; Microsoft Research, ftp://ftp.research.microsoft.com/pub/dtg/david/tutorial.ps

[Honeynet] www.honeynet.org

[Joyce] Rick Joyce & Gopal Gupta; *Identity Authentication Based on Keystroke Latencies*; CACM, 33(2) [February, 1990]

[Kitchens] Fred Kitchens, Sushil Sharma, & Queen Booker; *Identity Authentication Based on Keystroke Latencies Using the Genetic Adaptive Neural Network*

[Kohonen] T. Kohonen; *Self-Organizing Formation of Topologically Correct Feature Maps*; Biological Cybernetics; 43 [1982]

[Kontogianos] K. Kontogianos; *Evaluation Experiments on the Detection of Program Patterns Using Software Metrics*; 4th Working Conference on Reverse Engineering [1997]

[Krinke] Jens Krinke; *Identifying Similar Code with Program Dependence Graphs*; 8th Working Conference on Reverse Engineering [2001]

[Krsul] Ivan Krsul & Eugene Spafford; *Authorship Analysis: Identifying the Author of a Program*; Tech Report TR-96-052; Department of Computer Science, Purdue University [1996]

[Lammers] Angela Lammers & Sharon Langenfeld; *Identity Authentication Based on Keystroke Latencies Using Neural Networks*; Consortium for Computing in Small Colleges [1991]

[Lee-DPF] W. Lee and K. Park; *On the Effectiveness of Route-based Packet Filtering for Distributed DoS Attack Prevention in Power-law Internets*; http://www.cs.purdue.edu/nsl/recent-pubs.html/dpfsigcomm01.ps.gz; from proceedings at SIGCOMM, ACM Press, pp. 15 26 [2001]

[Lee-PPM] W. Lee and K. Park; *On the Effectiveness of Probabilistic Packet Marking for IP Traceback Under Denial of Service Attack*; http://www.ieee-infocom.org/2001/paper/721.ps; from proceedings at IEEE INFOCOM, IEEE CS Press [2001]

[Li] J. Li, J. Mirkovic, M. Wang, P. Reiher, and L. Zhang; *SAVE: Source Address Validity Enforcement Protocol* http://www.cs.ucla.edu/%7Esunshine/publications/ucla_tech_report_010004.pdf; from proceedings of IEEE INFOCOMM 2001 [April 2001]

[Liu] Peng Liu & Wanyu Zang; *Incentive-Based Modeling and Inference of Attacker Intent, Objectives, and Strategies*; CCS 03, Washington, D.C. [2003]

[Lustig] Michael Lustig & Yonit Shabtai; *Keystroke Attack on SSH*; Technical Report from Technion University, Israel [2001]

[Miller] Benjamin Miller; *Vital Signs of Personal Identity*; IEEE Spectrum [February 1990]

[Monrose] Fabian Monrose & Aviel Rubin; *Keystroke Dynamics as a Biometric for User Authentication*; Elsevier Press

[Moore] Andrew Moore, Robert Ellison, & Richard Linger; *Attack Modeling for Information Security and Survivability*; Technical Note SEI-2001-TN-001; Software Engineering Institute, Carnegie-Mellon University

[Morrow] Chris Morrow; *Black Hole Route Server and Tracking Traffic on an IP Network*; http://www.secsup.org/Tracking/; UUNET, WorldCom, Inc.

[MSS] MSS Initiative; http://mss.phildev.net/

[NetCamo] NetCamo Web site; http://research.cs.tamu.edu/netcamo/

[Ning] Peng Ning & Dingbang Xu; *Learning Attack Strategies From Intrusion Alerts*; CCS 03, Washington D.C. [2003]

[Onion Routing] Onion Routing Web site; http://www.onion-router.net/

[Paxson] V. Paxson; *An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks*; http://www.icir.org/vern/papers/reflectors.CCR.01/reflectors.html; Computer Communication Review, 31(3) [2001]

[Rabiner] Lawrence Rabiner; *A Tutorial on Hidden Markov models and Selected Applications in Speech Recognition*; from proceedings of the IEEE, 77(2) [February 1989]

[Reiger] Chuck Rieger; *An Organization of Knowledge for Problem Solving and Language Comprehension*; Artificial Intelligence, 7(2) [1976]

[Retinger] Philip Reitinger; *The Nature of the Threat*; IT Business Forum, WCIT 2000 [2000]

[RFC1812] F. Baker, Editor; *Requirements for IP Version 4 Routers*; http://www.ietf.org/rfc/rfc1812.txt [June 1995]

[RFC2827] P. Ferguson and D. Senie; *Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing*; ftp://ftp.isi.edu/in-notes/rfc2827.txt [May 2000]

[RFC2923] K. Lahey; *TCP Problems with Path MTU Discovery*; ftp://ftp.isi.edu/in-notes/rfc2923.txt [September 2000]

[Ribler] Randy Ribler & Marc Adams; *Using Visualization to Detect Plagiarism in Computer Science Classes*; IEEE Symposium on Information Visualization [2000]

[Romig] Steve Romig; *Forensic Computer Investigation*; presentation via http://www.acm.org

[Sallis] Philip Sallis, Asbjorn Aakjer & Stephen MacDonell; *Software Forensics: Old Methods for a New Science*; International Conference on Software Engineering: Education & Practice [1996]

[Savage] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson; *Practical Network Support for IP Traceback*; http://www.cs.washington.edu/homes/savage/papers/Sigcomm00.pdf; ACM SIGCOMM [August 2000]

[Snoeren] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer; *Single-packet IP Traceback*; http://www.net-tech.bbn.com/projects/SPIE/pubs/ton02.html; IEEE/ACM Transactions on Networking (ToN), 10(6) [December 2002]

[Schneier] Bruce Schneier; *Attack Trees: Modeling Security Threats*; Dr. Dobb's Journal [December 1999]

[Shanmugasundaram] Kulesh Shanmugasundaram; *Data Mining for Security Applications*

[SilentRunner] Now called eTrust Forensics, SilentRunner was acquired by Computer Associates; http://www.silentrunner.com

[Soboroff] Ian Soboroff, Charles Nicholas, James Kukla, & David Ebert; *Visualizing Document Authorship Using N-Grams and Latent Semantic Indexing*; NPIV 1997, Las Vegas, Nevada [1997]

[Song] ; Dawn Song, Peter Venable, & Adrian Perrig; *User Recognition by Keystroke Latency Pattern Analysis*; Carnegie Mellon University [1997]

[Song] Dawn Song and Adrian Perrig; *Advanced and Authenticated Marking Schemes for IP Traceback*; http://www.ece.cmu.edu/%7Eadrian/projects/iptraceback/tr-iptrace.pdf; IEEE Infocom 2001 [2001]

[Spafford] Eugene Spafford; *The Internet Worm: An Analysis; Computer Communication Review*, 19(1) [January 1989]

[Staniford] S. Staniford-Chen and L.T. Heberlein; *Holding Intruders Accountable on the Internet*; http://seclab.cs.ucdavis.edu/papers/thumb.ieee95.pdf; from proceedings of the 1995 IEEE Symposium on Security and Privacy [1995]

[Stone] Robert Stone; Center Track: *An IP Overlay Network for Tracking DoS Floods*; http://www.usenix.org/publications/library/proceedings/sec2000/stone.html; from proceedings of the 9th USENIX Security Symposium, Denver, Colorado [August 2000]

[Suzuki] Izumi Suzuki, Yoshiki Mikami, & Ario Ohsato; *A Language and Character Set Determination Method Based on N-Gram Statistics*; ACM Transactions on Asian Languages Processing, 1(3) [September 2002]

[SVM] Support Vector Machine tutorials may be found at: http://www.support-vector.net

[Telegraph] www.wsta.org/publications/articles/1003_article06.html

[Waldvogel] Marcel Waldvogel; *GOSSIB vs. IP Traceback Rumors*; http://marcel.wanda.ch/Publications/waldvogel02gossib.pdf; from proceedings at 18th Annual Computer Security Applications Conference (ACSAC 2002) [2002]

[Wang01] X. Wang, D. Reeves, S.F. Wu, and J. Yuil; *Sleepy Watermark Tracing: An Active Network-based Intrusion Response Framework*; http://seclab.cs.ucdavis.edu/papers/2001-03-watermark-ifipsec.pdf; from proceedings of IFIP Conference on Security, Paris, France [2001]

[Wang03] X. Wang and D. S. Reeves, *Robust Correlation of Encrypted Attack Traffic Through Stepping Stones by Manipulation of Interpacket Delays*; http://seclab.cs.ucdavis.edu/papers/2001-03-watermark-ifipsec.pdf; from proceedings of ACM Conference on Computer and Communications Security (CCS 2003) [2003]

[Weeber] Stephen Weeber & Eugene Spafford; *Software Forensics: Can We Track Code to Its Authors?;* Computers & Security, Volume 12, Number 6 [December 1993]

[Yaar] A. Yaar, A. Perrig and D. Song; *Pi: A New Defense Mechanism Against IP Spoofing and DDoS Attacks*; http://www.ece.cmu.edu/%7Eadrian/projects/pi.pdf; 2003 IEEE Symposium on Security and Privacy [2003]

[Zhang] Yin Zhang & Vern Paxson; *Detecting Stepping Stones*; http://www.usenix.org/publications/library/proceedings/sec2000/full_papers/zhangstepping/zhangstepping.pdf; 9th USENIX Security Symposium, Denver, Colorado [August 14-17, 2000]