

A TWO-LAYER DETERMINISTIC HIERARCHICAL MARKING
SCHEME FOR IP TRACEBACK

MASTERS PROJECT

Submitted to

Dr. Nael B. Abu-Ghazaleh
Assistant Professor
Binghamton University
Department of Computer Science
Binghamton, NY 13902-6000

By

Christian Bongiorno
M.S., Computer Science
Binghamton University, Binghamton NY, USA
2004

ABSTRACT

This work investigates a technique for tracing IP packets back to their source even when only a trickle of packets is available. The proposed approach divides the problem into two levels: (1) host traceback – determining which host within the domain originated the packet and (2) originating network traceback – determining the administrative domain that originated the packet. We further take into consideration Network Address Translation (NAT) and how it affects other *traceback* schemes

To solve our first problem, we propose marking the packet deterministically at the ingress router. To solve our second problem, we further mark the packets at the ingress LAN-AS boundary. In packet marking schemes, it is essential to ensure that the attacker cannot circumvent the traceback by spoofing packet marking information. We consider several possible attacks and propose solutions to them. We assert that our proposed approach can be incrementally deployed, requires minimal overhead, and truly can traceback to the origin host, all while maintaining backward compatibility.

TABLE OF CONTENTS

ABSTRACT.....	2
LIST OF FIGURES.....	4
1.0 INTRODUCTION	5
2.0 RELATED WORK	8
2.1 Packet Marking	8
2.1.1 Probabilistic Packet Marking	8
2.1.2 Deterministic Packet Marking	11
2.2 Router Based Approach	13
2.3 Out-of-band Approaches	15
2.4 Trace-back of Active Attack flows	15
2.5 Other Approaches	16
3.0 OVERVIEW OF PROPOSED SOLUTION	18
3.1 Assumptions on Attacker Capabilities	19
3.2 Basic Algorithm	20
3.2.1 Marking the Host	21
3.2.2 Generating Network Mark.....	22
3.2.3 Addressing areas of concern.....	23
3.3 Data requirements per packet.....	24
3.3.1 Creating space in each packet.....	26
3.3.2 Anti-fragmentation procedure.....	28
4.0 EXPERIMENTATION AND ANALYSIS.....	30
4.1 Our Traceback approach in a simple topology.....	30
4.2 Our traceback approach through multiple ASs with a NATed stub network.....	31
4.3 Our approach successfully marking a spoofed packet.....	34
4.4 Traceback in Mixed Transit-leaf Segments.....	35
4.5 Traceback on a Pure Transit Segment.....	37
4.6 Limitations.....	39
5.0 IDEAS FOR THE FUTURE AND CONCLUSIONS.....	40
5.1 Future work	40
5.2 Conclusions	41
REFERENCES	45

LIST OF FIGURES

Figure 1 Host Marking.....	19
Figure 2 Network Marking	21
Figure 3 Attacker on Transit Segment.....	23
Figure 4 The Network Traceback Header.....	26
Figure 5 A Mixed Mode Segment.....	35
Figure 6 A Pure Transit Segment	37

1.0 INTRODUCTION

Network security breaches represent a growing threat to businesses and institutions, costing them billions of dollars every year. The datagram nature of the Internet makes it difficult to determine the originating host of a packet – the source id supplied in an IP packet can be falsified (IP spoofing). The problem of finding the source of a packet is called the *IP traceback* problem. *IP Traceback* is a critical ability for identifying sources of attacks and instituting protection measures for the Internet. Most existing approaches to this problem have been tailored toward DoS attack detection. Such solutions require high numbers of packets (tens of thousands) to converge on the attack path(s). More importantly, these approaches attempt to track the path from attacker to victim; this work holds that such information is extraneous and complicates the matter substantially.

These approaches also do nothing to defend against the nefarious “hacker” that has plagued the Internet since its infancy. An attack initiated by such a “hacker” would come in trickles of packets and not floods, making such approaches ineffective.

In this work, we propose an alternative approach to IP traceback that breaks the problem into two parts: (1) host traceback and (2) originating network traceback

In the first part, we seek to identify a packet within a flat limited size topology. The approach we take is to mark each packet at the ingress router connecting the originating

network to its service provider. For the second part, having identified the originating host, we seek to identify the network that originated the attack packet. Our approach here is try and mark the packet as close to the originating host as possible.

A conundrum does arise in our approach. Essentially, we have 3 different types of interfaces that may be connected to a routing device: A leaf node (an end host), a transit link, or a mixed leaf-transit link. On a leaf-link we always mark the packet. Unfortunately, in a mixed network or transit network we have no way of truly distinguishing if a packet should be marked or not without further devices. We discuss maneuvering around this problem later.

There are two possible attacks on our *traceback* approach:

- 1) spoofing: a host may falsify any information in a packet.
- 2) The attacker may own or have subverted a router close to it: In such a case, it may use this router to spoof marking information.

We develop and study solutions to both of these attacks.

In our experiments, we show that simple deterministic *traceback* is indeed feasible. That is, with no attacks and no NAT, we can truly identify the source of the packet without regard for IP address.

We also show that, given a stub network using NAT, any *traceback* compliant routing device effectively traces the packet back to entry point of the NATed stub network. Further, we show that even without *ingress filtering* we can accurately *traceback* a spoofed packet to its origin – even if it steals the IP address of a valid neighbor on the subnet (to defeat *ingress filtering*).

Next, we show that in mixed transit-leaf networks or pure transit, we are able to once again accurately *traceback* the packet for spoofed and non-spoofed packets – even those originating from inside the backbone. Most importantly, we provide a mechanism for effectively dealing with the possibility that a packet may contain falsified marking information.

From our experiments, it is clear that this approach will work and can be deployed in a hierarchical fashion to better facilitate deployment. Performance overhead from using the options was not evaluated, but there is evidence (to be discussed later) from other areas of networking that suggests that this likely will not be a problem.

The remainder of this report is organized as follows. In Section 2, we overview some related approaches. Independently of this research, others have proposed deterministic *traceback* solutions as well. Section 3 overviews the proposed *traceback* scheme. Section 4 presents some experiments and analysis. Finally, Section 5 presents some ideas for future work and concluding remarks.

2.0 RELATED WORK

A brute force solution to traceback can be obtained by having every router mark every packet as it passes through it. An alternative brute force solution requires every router to keep a record of every packet that passes through it. Such solutions are infeasible because: (1) they require a large and unbounded space in each packet (or router); (2) they require a large overhead at every router. Thus, most existing approaches to traceback attempt to reduce the above two effects.

The discussion of the related work is organized as follows: (1) packet marking: packets are marked with traceback information – of this type of scheme, there are essentially 2 differing types: probabilistic and deterministic; (2) router marking: routers maintain records of packets that pass through them; (3) out-of-band marking: new packets are generated to inform a destination that a packet destined to it was seen; (4) Active Flow tracking; and (5) Other approaches.

2.1 Packet Marking

2.1.1 Probabilistic Packet Marking

Savage et al¹ suggest probabilistically marking packets as they traverse routers in the Internet. More specifically, they propose that router mark the packet, with low probability (say, 1/20,000), with either the router's IP address or the edges of the path that the packet traversed to reach the router.

For the first alternative, analysis shows that in order to gain the correct attack path with 95% accuracy as many as 294,000 packets are required [1]. The second approach, edge marking, requires that the two nodes that make up an edge mark the path with their IP addresses along with a distance of 8 bits to represent the maximum hop count allowable in IP. This approach would require more state information in each packet than simple node marking but would converge much faster. They suggest 3 ways to reduce the state information of these approaches into something more manageable.

The first approach is to XOR each node forming an edge in the path with each other. Node a inserts its IP address into the packet and sends it to b. Upon being detected at b (By detecting a 0 in the distance), b XORs its address with the address of a. This new data entity is called an edge id and reduces the required state for edge sampling by half.

Their next approach is to further take this edge id and fragment it into k smaller fragments. Then, randomly select a fragment and encode it, along with the fragment offset so that the correct corresponding fragment is selected from a downstream router for processing.

When enough packets are received, the victim will receive all edges and all fragments so that an attack path can be reconstructed (even in the presence of multiple attackers). The low probability of marking reduces the associated overheads. Moreover, only a fixed space is needed in each packet.

Due to the high number of combinations required to rebuild a fragmented *edge id*, the reconstruction of such an attack graph is computationally intensive according to research by Song and Perrig ². Furthermore, the approach results in a large number of false positives. As an example, with only 25 attacking hosts in a DDoS attack the reconstruction process takes days to build and results in thousands of false positives[2].

Accordingly, Song and Perrig propose the following traceback scheme: instead of encoding the IP address interleaved with a hash, they suggest encoding the IP address into a 11 bit hash and maintain a 5 bit hop count, both stored in the 16-bit fragment ID field. This is based on the observation that a 5-bit hop count (32 max hops) is sufficient for almost all Internet routes. Further, they suggest that two different hashing functions be used so that the order of the routers in the markings can be determined. Next, if any given hop decides to mark it first checks the distance field for a 0, which implies that a previous router has already marked it. If this is the case, it generates an 11-bit hash of its own IP address and then XORs it with the previous hop. If it finds a non-zero hop count it inserts its IP-hash, sets the hop count to zero and forwards the packet on. If a router decides not to mark the packet it merely increments the hop count in the overloaded fragment id field.

Song and Perrig [2] identify that this is not robust enough against collisions and thus suggest using a set of independent hash functions, randomly selecting one, and then hashing the IP along with a *FID* or function id and then encoding this. They state that this

approach essentially reduces the probability of collision to $(1/(2^{11})^m)$. For further details see Song and Perrig [2].

2.1.2 Deterministic Packet Marking

Belenky and Ansari³, with an approach nearly identical to the one taken in this paper, outlined deterministic packet marking initially. In their letter they describe a more realistic topology for the Internet – that is composed of LANs and ASs with a connective boundary – and attempt to put a single mark in the packet on inbound packets at the point of network ingress. Their idea is to put, with random probability of .5, the upper or lower half of the IP address of the ingress interface into the fragment id field of the packet, and then set a reserve bit indicating which portion of the address is contained in the fragment field. By using this approach they claim to be able to obtain 0 false positives with .99 probability after only 7 packets.

Unfortunately, their approach is extremely light on details and fails to address how upstream routers are to account for their marking if they too are also marked. They also fail to discuss trans-leaf segments, shared segments and transit segments. Also, their approach does not take into account the non-unique status of an IP address that NAT confers on network topologies. No follow up work to this paper has been discovered.

Rayanchu and Barua⁴ provide another spin on this approach (called DERM). Their approach is similar in that they wish to use and encode IP address, of the input interface, in the fragment id field of the packet. Where they differ from Belenky and Ansari is that

they wish to encode the IP address as a 16-bit hash of that IP address. Initially they choose a known hashing function. They state that there would be some collisions if there were greater than 2^{16} edge routers doing the marking. In truth, it need not require so many to achieve a collision (by nature of probabilities); this collision space is further increased by the fact that inbound interface address are used and any “router” will very likely have multiple interfaces; all with differing IP addresses.

They attempt to mitigate the collision problem by introducing a random distributed selection of a hash function from the universal set, and then applying it to the IP address (while a hash identifier is discussed, how it is communicated with the receiver is unknown). In either hashing scenario, the source address and the hash are mapped together in a table for later lookup along with a bit indicating which portion of the address they have received. Through a complicated procedure and a random hash selection, they are capable of reducing address collision.

By using a deterministic approach they reduce the time for their reconstruction procedure for their mark (the 16 bit hash). However, by encoding that mark through hashing they introduce the probability of collisions, and thus false-positives. As a parallel, just like Belenky and Ansari, their approach does not address NAT – a significant shortcoming.

2.2 Router Based Approach

In this type of solution, the router is charged with maintaining information regarding packets that pass through it. For example, Sager⁵ proposes to log packets and then data mine them later. This has the benefit of being out of band and thus not hindering the fast path. However, given link speeds on core routers, this simply will not scale. Tera-bytes would be required for logging and even if that much space were used, the sheer I/O demand on the disks would slow everything down. Of course, probabilistic sampling techniques can reduce the data demands for this scheme.

Snoeren et al⁶ propose marking within the router. The idea proposed in their paper is to generate a *fingerprint* of the packet, based upon the invariant portions of the packet (source, destination, etc.) and the first 8 bytes of payload (which is unique enough to have a low probability of collision). More specifically, m independent simple hash functions each generate an output in the range of 2^{n-1} . A bit is then set at the index generated to create a *fingerprint* when combined with the output of all other hash functions. All *fingerprints* are stored in a 2^n bit table for later retrieval. The paper shows a simple family of hash functions suitable for this purpose and present a hardware implementation of it.

The space needed at each router is limited and controllable (2^n bits). A small n makes the probability of collision of packet hashes (and false identification) higher. When a packet is to be traced back, it is forwarded to originating routers where fingerprint matches are checked. As time passes, the fingerprint information is “clobbered” by hashes generated

by other packets. Thus, the selectivity of this approach degrades with the time that has passed between the passage of the packet and the traceback interrogation. A large drawback of this approach is that it requires significant changes to router infrastructure – true for any packet-marking scheme.

The final known take on the router-based schemes comes from Hiroaki Hazeyama et al⁷. In their approach, they wish to integrate the SPIE approach as outlined by Snoeren [6] (above), with their approach of recording the layer 2 link-id along with the network ID (VLAN or true ID), the MAC address of the layer 2 switch that received the packet and the link id it came in on. This information is then put into two lookup tables – both containing the switch (layer 2 router) MAC id for lookup. They rely on the MAC:port tuple as a method of tracing a packet back (even if the MAC address has been spoofed).

This mechanism certainly could work, but there is a finite space on any given switch to store such information. To help mitigate this problem, they use Snoeren's hashing approach and implementation (SPIE) – modifying it to accept their information for hashing. By their own admission, their algorithm is slow ($O(N^2)$) and, with only 3.3 million packet hashes being stored the approximate time before the digest tables are invalid is 1 minute. This dictates that any attack response must be real-time – a possibility only on single-administrative LAN domains. To their credit though, they are using real hardware implementations of their idea – which alleviates some arguments about plausibility.

2.3 Out-of-band Approaches

The ICMP traceback scheme Bellovin⁸ proposes probabilistically sending an ICMP *traceback* packet forward to the destination host of an IP packet with some low probability. Thus, the need to maintain state in either the packet or the router is obviated. Furthermore, the low probability keeps the processing overhead as well as the bandwidth requirement low. Bellovin suggests that the selection also be based on Pseudo-random numbers to help block attempts to time attack bursts. The problem with this approach is that routers commonly block ICMP messages because of security issues associated with them.

2.4 Trace-back of Active Attack flows

In this type of solution, an observer tracks an existing attack flow by examining incoming and outgoing ports on routers starting from the host under attack. Thus, such a solution requires having privileged access to routers along the attack path.

To bypass this restriction and automate this process, Stone⁹ proposes routing suspicious packets on an overlay network using ISP edge routers. By simplifying the topology, suspicious packets can easily be re-routed to a specialized network for further analysis.

This is an interesting approach. By nature of DoS, any such attack will be sufficiently long lived for tracking in such a fashion to be possible. Layer-three topology changes, while hard to mask to a determined attacker, have the possibility of alleviating the DoS until the routing change is discovered and subsequently adapted to. Once the attacker has

adapted, the re-routing scheme can once again adapt and re-route; causing an oscillation in the DoS attack; granting some ability to absorb the impact of such an attack.

2.5 Other Approaches

Burch and Cheswick¹⁰ propose a controlled flooding of links to determine how this flooding affects the attack stream. Flooding a link will cause all packets, including packets from the attacker, to be dropped with the same probability. We can conclude from this that if a given link were flooded, and packets from the attacker slowed, then this link must be part of the attack path. Then recursively upstream routers are “coerced” into performing this test until the attack path is discovered. The most important problem with this approach is that it is resource intensive and highly intrusive. In fact, this approach may be viewed itself as a DoS attack.

The traceback problem is complicated because of spoofed packets. Thus, a related effort is targeted towards preventing spoofed packets; known as *Ingress Filtering*. *Ingress Filtering* restricts spoofed packets at ingress points to the network by tracking the set of legitimate source networks that can use this router.

Park and Lee¹¹ present an extension of *Ingress Filtering* at layer 3. They present a means of detecting false packets, at least to the subnet, by essentially making use of existing OSPF routing state to have routers make intelligent decisions about whether or not a packet should be routed. Their approach makes sense and can be implemented inside of routers with little overhead.

Ingress filtering is recommended IETF practice and is widely deployed. However, a sophisticated attacker can still spoof packets from the set of networks that its Ingress Filter allows. This is especially true for networks that are used for transit, where the set of legal sources may be large. Ingress filtering extensions down to the leaf networks have been proposed to further limit spoofing. Regardless, by blocking packets with known illegitimate addresses at routing devices, the space of the attack origin is substantially reduced.¹²

3.0 OVERVIEW OF PROPOSED SOLUTION

The United States Post Service (USPS) has been in the business of delivering “packets” much longer than the Internet has. The problem of anonymous sources, whether intentionally or not, was dealt with long ago at the post office by placing an overlay stamp on the postage indicating which zip code the parcel originated from. This information could be trusted to be authentic since it was placed there by the post office, which presumably is not compromised by someone trying to send an anonymous letter.

Existing solutions to the *traceback* problem attempt to reduce the state and processing overheads by a combination of probabilistically generating traceback information, and/or using hash functions to reduce the marking state. As such, for these to be effective *traceback* requires a colossal number of packets, and vast computing resources for reconstructing the attack graph.

Many of these solutions ultimately assume that the Internet is a Directed Acyclic Graph (*DAG*). In truth, the Internet as a whole is not acyclic and further, knowing the entire attack path is irrelevant information – what is truly coveted is the source of the packet. Only a fraction of how it got there is useful.

In this work, we propose an alternative traceback scheme that deterministically marks packets at well-specified points in the network. More specifically, we seek to mark the packet as close as possible to its origin. Indeed, at the very port and switch it came from. Such a mark is sufficient to traceback an attacking packet to its origins without complex

post processing. The marking can be carried out probabilistically if desired (using works discussed in the prior art section), but limiting the points of marking results in constant state overhead and allows aggressive marking (every packet may be marked). Thus, even non-DoS attacks can be tracked successfully to their origin.

The proposed approach is a two-level hierarchical marking scheme that takes advantage of the organization of the Internet. The approach consists of:

- 1) Marking the packet within a customer network (*host mark*). Being within a single administrative domain makes *traceback* a more manageable problem. and
- 2) Marking the packet at routers connecting customer networks to provider networks (*network mark*). The underlying assumption here is that such routers are not compromised.

We explore several possible attacks on our scheme and propose mechanisms to defeat them.

3.1 Assumptions on Attacker Capabilities

- An attacker may generate a packet of any form
- Attackers are aware that they are being traced
- Multiple attackers may conspire
- Routers and switches are not widely compromised

Any compromised router along an attack path would be able to eliminate marking information from upstream routers. This is a vulnerability of all packet-marking

schemes. However, there are valid reasons to make the assumption on router integrity especially beyond the immediate attacker LAN. Primarily, if a router somewhere along the Internet were to be compromised (say a backbone router in AT&T) then the security threat that exists from such a compromise far outweighs any problem associated with traceback.

3.2 Basic Algorithm

We first begin by identifying the Internet not as a flat graph. Instead, it consists of independently administered networks that are interconnected through edge routers (Figure 2). In this figure, we use the term Autonomous System (AS) to indicate an independently administered network; however, note that a true AS (in the BGP sense) may include several independently managed networks (such as corporate LANs or households using NAT). The backbone refers to one or more ISP networks.

3.2.1 Marking the Host

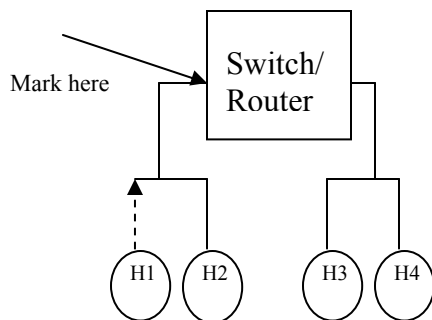


Figure 1: Host Marking

To begin with, an IP packet generally traverses several switches and internal routers before it leaves an AS. We seek to apply the marking on a packet at the point where it

enters the network. An ingress switch or router can be manually configured to start marking or judicious analysis of OSPF routing data could signal to the routing device that it is on a leaf network and thus should begin marking.

Conceding that the attacker's machine is compromised, the first logical point to mark the packet is the first switch/router on the LAN (Figure 1). The switch marks the packet with information identifying the switch (Switch ID, or SID) and the port the packet came on; we call these two pieces of information together the host mark. The SID could be the MAC id of the switch or any other information that can be locally identified.

Ideally we would like to stay away from IP addresses for such marking purposes. They can be dynamic due to DHCP leasing, the IP address may not necessarily be unique between different sites (due to the common use of non-routable IP addresses in NAT environments), and because the MAC address is globally unique; allowing for the ability to actually identify the vendor and possible owner of the routing device.

If such information can be maintained in a packet, then a victim can query real-time or post-mortem for who is connected to that switch. It is important to note that disclosing the MAC id of a switch might pose a security risk for the same reason it is also useful; that is, knowing the vendor by MAC id may give an attacker insight as to vulnerabilities that might exist. As such, great cogitation on the origin of the SID should occur. For purposes of this study the SID will be assumed to be a 48-bit Ethernet MAC address.

Given that a packet can have its source address spoofed, the AS with which this switch belongs to cannot be determined by looking at the source IP address and thus, it is also necessary to have a trusted entity report from which AS a given packet came from.

In-and-of itself this is not necessarily such a drawback for LAN environments. A query for *traceback* information on a LAN can be orders of magnitude faster, whether automated or by physically contacting an administrator for such information, than if such a query had to occur across the Internet. It should also be noted that the majority of system compromises occur inside a secured network and not from external Internet sources.¹³

3.2.2 Generating Network Mark

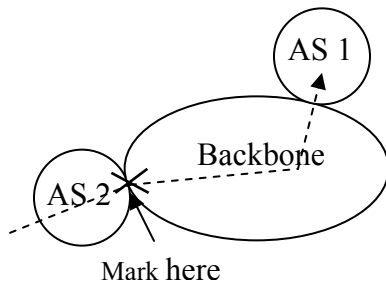


Figure 2: Network Marking

We observe next that, as a packet travels from a host in AS1 it is handed off from the AS to the backbone and then to the destination AS. It is when a packet attempts to leave the origin AS that we wish to mark it.

To this end, such information is readily available in BGP¹⁴. Routing in BGP is done between ASs and as such, a backbone router could be used to mark which AS any given

packet came from. This AS ID will ultimately narrow down the source of the packet with a given SID.

From the network mark, we can isolate the originating network of a packet and from the host mark we can identify the host or LAN segment of the attack origin. By simply marking at the appropriate points, we have a finite storage need and a deterministic *traceback*.

3.2.3 Addressing areas of concern

There is a problem area with this approach that requires a special solution. An ingress switch/router cannot discriminate between leaf host packets (which should be marked) and transit packets (which should not). This is a problem at the LAN level and the AS level. The decision to mark or not breaks itself down into 3 distinctive routing categories:

- Pure leaf segments
- Pure transit segments
- Mixed transit and leaf

As an example, consider the following attacks on this scheme:

- 1) Attacker spoofs the host mark in an attempt to prevent the switch from marking it on a transit LAN.
- 2) A related issue is single segment attacks where the attacker resides on the same segment as the victim and the router does not have a chance to mark.
- 3) Attacker subverts a router in the attack path. (See figure 3)

4) Alternatively, a non-traceback enabled routing device exists along the route.

On leaf (non-transit) LANs, any manipulation of the packet cannot succeed since the switch deterministically marks every packet leaving the LAN segment. In transit and mixed mode, we have two options: mark every packet or discriminate them. Marking every packet would lose us granularity, and at the AS level, losing host level granularity on a non-DoS attack would essentially render this method useless.

To solve these problems, first we use ingress filtering at the first router to limit the set of sources that may forward packets through each port of a switch. For most leaf routers, identifying the legal sources is trivial (a single subnet prefix). Moreover, Park and Lee's approach for packet filtering based on routing information is suitable here even for transit LAN segments [11].

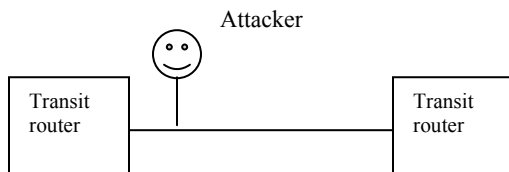


Figure 3 Attacker on a Transit Segment

With such a mechanism in place, spoofing will not be effective unless the attacker spoofs the address of another legal source. Unfortunately, if the LAN segment is a transit segment, the set of legal sources may be large.

Essentially, the solution represents an authentication of the transit path. Fortunately for all, such authentication mechanisms exist – the one that comes to mind first is IPsec. Using IPsec we are able to get around the 2 above routing problems.

First, by using IP sec, if we have a compromised router at *B* between *A* and *C* (or a non-traceback compliant router), then the original marked packet is tunneled to the next trusted router in the path. If it is believed that the router is compromised, then the packet can also be encrypted. In the case of a mixed mode transit-leaf segment or pure transit, authentication-using IPsec can either be done between transit routers (less scalable) or between ingress switch (larger deployment barrier) and transit router. In either case, the packet to be left unmolested is immediately distinguishable.

This gracefully returns us to a prior point of discussion: The amount of overhead our approach requires. The IPsec RFCs specify numerous means of reducing computing costs in regards to encryption and authentication. Also, a recent and unrelated problem with TCP may give some insight as to how much computing resources are truly available on any high-speed router.

The recent TCP RST exploit as explained by Watson¹⁵ has caused a flurry of discussion to prevent this attack against BGP. Relevant to this discussion, Watson [15] recommends using the MD5 authentication mechanism as specified in RFC 2385 to protect against spoofed packets.

It is on this point that some experts believe MD5 hashing to authenticate routing information does not pose a serious computational problem¹⁶. MD5 is one of the hashes discussed for use in IPSec.

3.3 Data requirements per packet

3.3.1 Creating space in each packet

Essentially then, each packet must be marked with the AS id, the SID and the port number that the packet came in on. The size of all this information would be a 16-bit AS ID, a 48-bit SID, and a 6-bit port id (we assume most switch have less than 64 ports).

This makes for a 70 bit spacing requirement in IP. With only minimal compression (such as dropping the high-order byte of the MAC, which is typically '00') we can get the culmination of this data to fit on a 32-bit boundary. The tricky part now is how to fit these 70-bits of data into an already packed IP header without fragmentation. There are a few ways this can be done with trade-offs of each approach to be considered. Before we discuss methods of accommodating this new information on a MAC frame level, we make a brief detour to discuss how we plan to modify the IP packet to suite our needs.

Quite simply, the AS:SID:PORT *traceback* information is to be put into the options field. We leave it up to the industrious reader to contrive a method for reducing its space requirements. (See figure 3)

To implement such a packet modification in an ingress switch would not be challenging. The incoming packet header could be copied into a small buffer; attaching the *traceback* information to the end of that buffer, and finally the original payload to the end of *traceback* information.

We then set one of the reserve bits in the TOS field (bit-4 High Throughput), recompute and update the IP checksum and length field in our new packet, and then send the packet back into the main processing stream/buffer so that the packet is further treated normally (even fragmented if necessary).

In case the reader is wondering, we set a TOS bit to allowing for hops along the way to identify the extra options in the packet as **IP_TRACEBACK**; thus, staying on the fast path and avoiding the need to bring packets up to the software layer to handle options. It is crucial that a method exist for staying on the fast path; without it, this could bring a high-speed routing device to its knees.

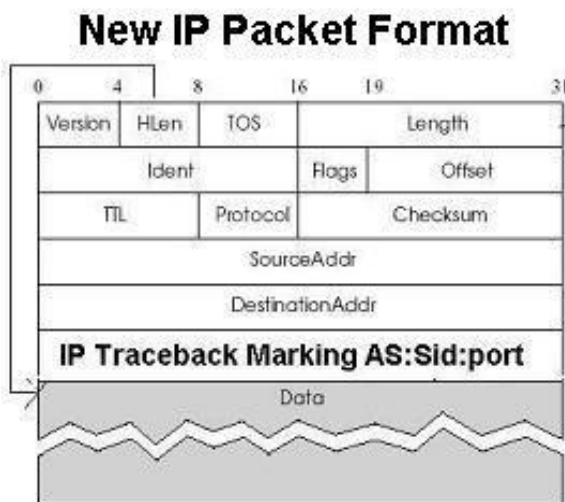


Figure 4 – the New *Traceback* header

Now that the structure of the new packet is defined, accommodations for it in a frame will be made to prevent fragmentation.

3.3.2 Anti-fragmentation procedure

To accommodate our *traceback* information we need to be able to reserve room at the end host in every packet it transmits. This can be done in one of two ways, either at the first switch, or the end host.

To update the end host, we could deploy a slightly modified protocol stack that simply adjusts the frame size internally to make room for traceback – software updates to an OS are routine and usually free.

The other approach would be to simply have the traceback switch actually provide an adjusted MTU when it sees a MTU discovery attempt. Modern network stacks implement automatic MTU discovery to prevent fragmentation regardless of the underlying media.¹⁷ Therefore, having the traceback switch modify the ICMP “Datagram too big” message would require very little effort, would be highly transparent and could be part of the requirements for an IP Traceback enabled switch.

In the case that end hosts are non-compliant (whether their networks stacks are not compliant or such a host has been compromised) inserting our IP Traceback mark would cause minimal fragmentation. A “hacker” would only hurt himself from attempts to

modify the packet to his advantage. Further, such fragmentation could be used as a signature for intrusion detection

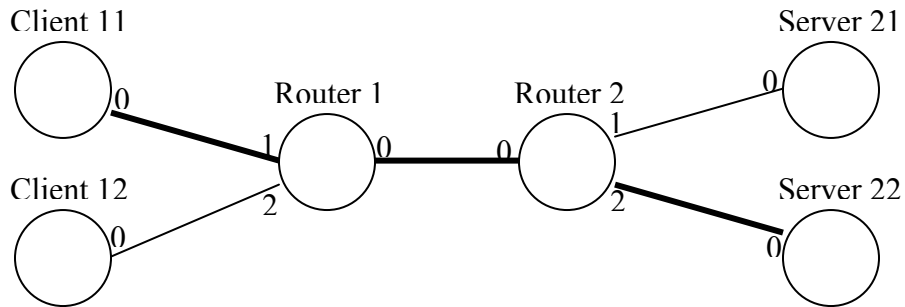
In any event, packets would be subject to fragmentation even if traceback were not enabled due to lack of MTU discovery. Like any solution, there are strengths and weaknesses that must be considered.

4.0 EXPERIMENTATION AND ANALYSIS

In this section, we highlight and discuss different configurations and outputs from simulating our approach using the SSFnet simulator under Java.

4.1 Our *Traceback* approach in a simple topology

This experiment is simple. It comprises 4 hosts and two routers; one router (router ID 1) is *iptraceback* enabled. We initiate a single connection (from host 11) and transfer some data from a client on the *traceback*-enabled network to a server (host 21) located behind router 2. The receiving server has a special IP stack installed on it that merely outputs our *traceback* information when it is present. Below is a diagram of the topology results; much of the output omitted for brevity:



```

CIDR          IP Block          b16          NHI
--           0.0.0.0/27      0x00000000
0            0.0.0.16/30     0x00000010   1 (0) 2 (0)
1            0.0.0.12/30     0x0000000c   1 (1) 11 (0)
2            0.0.0.8/30      0x00000008   1 (2) 12 (0)
3            0.0.0.4/30      0x00000004   2 (1) 21 (0)
4            0.0.0.0/30      0x00000000   2 (2) 22 (0)
NHI Addr          CIDR Level          IP Address Block    % util
--              --              0.0.0.0/27          68.75
**              Using specified 1.0n clock resolution

--- Phase III: add static routes
## Net config: 6 routers and hosts
## Elapsed time: 0.144 seconds
  
```

```

** Running for 5678900000000000 clock ticks (== 567890.0 seconds sim
time)
1.616458204 TCP host 11 src={0.0.0.14:10001} dest={0.0.0.6:1600}
Active Open
Host 21 PKT RCVD ID: 0 marked by: |1| AS:-1 SID:1 LINK:1
1.623823921 TCP host 21 src={0.0.0.6:1600} dest={0.0.0.14:10001} SYN
rcvd
Host 21 PKT RCVD ID: 2 marked by: |1| AS:-1 SID:1 LINK:1
...
Host 21 PKT RCVD ID: 19814 marked by: |1| AS:-1 SID:1 LINK:1
Host 21 PKT RCVD ID: 19816 marked by: |1| AS:-1 SID:1 LINK:1
10.02294903 [ sid 1 start 1.616458204 ] client 11 srv 21(0) rcvd
10000000B at 9516.455kbps - read() SUCCESS
...
Host 21 PKT RCVD ID: 19988 marked by: |1| AS:-1 SID:1 LINK:1
10.02294903 TCP host 11 src={0.0.0.14:10001} dest={0.0.0.6:1600}
Active Close
Host 21 PKT RCVD ID: 19990 marked by: |1| AS:-1 SID:1 LINK:1
...
-----
| 1 timelines, 5 barriers, 1195831 events, 17949 ms, 67 Kevt/s
-----
Process terminated with exit code 0

```

Looking at the above output, certain lines are shaded. This is the information dumped by our server with the special IP stack to show the *traceback* information received. AS:-1 SID:1 LINK:1 – in this implementation, when the AS id is -1, it means that the packet never crossed an AS boundary. In this configuration, no ASs are defined. SID 1 show that the packet entered the network on Switch ID 1. From our configuration, we can see that indeed that was the case. Lastly, Link 1 – the packet was received on switch ID 1, interface 1. Again, looking at the configuration we can see that indeed this is the case. From this simple and judicious configuration, we can see that our *traceback* mechanism does indeed correlate with the topology of this network; proving that our approach can indeed work (at least for the LAN level).

4.2 Our traceback approach through multiple ASs with a NATed stub network

In this simulation we show that, not only can we track across and AS boundary successfully, but also that indeed we can track a NATed packet back to its origin where

the stub meets the main network. This simulated network is large, and is the basis for the rest of our experiments, as it much more typical of what can be expected. So large infact, that to diagram it would be an exercise in tedium. We briefly describe the topology and allow the reader to explore the configuration further as desired (See the SSFnet distribution examples/ComboDemo2.dml).

This network is comprised of 6 backbone routers and 32 networks (each network being an AS unto itself). Of those 32 networks, half are all client networks (even numbered networks), and half are all server networks (odd numbered networks). In this scenario, we attach a stub NATed network (ID 999) to one of the routers in our client network (router ID 6). Since there are 16 duplicate client networks, there are 16 NATed stubs; only host 2:999:2 (network 2, subnet 999, host 2) is generating traffic bound for host 13, network 35 (35:13).

```

CIDR          IP Block          b16          NHI
--           0.0.0.0/20      0x00000000
2            0.0.8.128/25      0x00000880   2
2/0         0.0.8.128/27      0x00000880   2:999
2/0/0      0.0.8.140/30      0x0000088c   2:999:1 (1) 2:999:2 (0)
2/0/1      0.0.8.136/30      0x00000888   2:999:1 (2) 2:999:3 (0)
2/0/2      0.0.8.132/30      0x00000884   2:999:1 (3) 2:999:4 (0)
2/0/3      0.0.8.128/30      0x00000880   2:999:1 (4) 2:999:5 (0)

2/12       0.0.8.168/29      0x000008a8   2:6(2) 2:17(0) 2:18(0)
2/13       0.0.8.160/29      0x000008a0   2:6(1) 2:19(0) 2:20(0)
2/14       0.0.8.216/30      0x000008d8   2:6(4) 2:999:1(0)
...
35          0.0.9.192/26      0x000009c0   35
35/8       0.0.9.192/29      0x000009c0   35:3(1) 35:12(0) 35:13(0)
**          Using specified 1.0ns clock resolution
--- Phase I: construct table of routers and hosts

HOST: 2:999:1
PUBLIC ifaces [2266]
PRIV ifaces [2177, 2189, 2181, 2185]
Host 2 ->0.0.8.142
Host 3 ->0.0.8.138
Host 4 ->0.0.8.134

```

```

Host 5 ->0.0.8.130

--- Phase II: connect Point-To-Point links
--- Phase III: add static routes
## Net config: 741 routers and hosts
## Elapsed time: 4.764 seconds
** Running for 5679000000000000 clock ticks (== 56790.0 seconds sim time)
Host 35:13 PKT RCVD ID: 13130 marked by: |2:6| AS:3 SID:6 LINK:4
...
206.443906786 [ sid 1 start 203.882597511 ] client 2:999:2 srv 35:13(0)
rcvd 10000B at 31.234kbps - read() SUCCESS
Host 35:13 PKT RCVD ID: 13148 marked by: |2:6| AS:3 SID:6 LINK:4
Host 35:13 PKT RCVD ID: 13149 marked by: |2:6| AS:3 SID:6 LINK:4
Host 35:13 PKT RCVD ID: 13151 marked by: |2:6| AS:3 SID:6 LINK:4
-----
| 1 timelines, 5 barriers, 5169114 events, 39818 ms, 134 Kevt/s
-----
Process terminated with exit code 0

```

The first highlighted section under “*Phase I*” is output from our implementation of NAT. Here, it simply shows that we have a set of public interface (only 1, IP: 2266 in base 10), and private interfaces (4 of them, IPs: [2177, 2189, 2181, 2185]). Looking at the output from Phase III, we can see that Host 35:13 did indeed receive packets with our *traceback* information. Accordingly, that information states that it came from AS 3, Switch ID 6, on interface 4. We realized after some analysis that the BGP implementation used in SSFnet uses a 1-based counting scheme for AS numbers. Following the highlighted routing information in our above table, we can see that if indeed we followed this *traceback* information, it leads us back to the interface on which our NATed stub network is connected to the main one.

With this simulation we have shown several things: First, that we can successfully *traceback* to the origin AS of a packet. Secondly, that NAT does not interfere with the operation of *traceback* in anyway. In this case, we were unable to *traceback* into the stub network not technically because of NAT, but because this implementation of NAT is not *traceback* compliant. Finally, it’s this very example of NAT that shows us that we can

have a hierarchical approach to deployment and still receive some benefit. If this NATed network had more routers beneath it that were also not compliant, we would not lose traceback granularity from what we have now; alas, we would not gain any either

4.3 Our approach successfully marking a spoofed packet

Up until now, our simulations and their subsequent results have demonstrated that our approach to *traceback* works just fine when everyone one is playing nice-nice. In this simulation we show that, even when there is a maliciously host that is spoofing IP source addresses, we can still *traceback* the attacker successfully. We use the same topology as 4.2 only this time the client is not NATed and the client is maliciously spoofing the IP source address between 0 and 255. It is thus not surprising that our server receives ICMP DHU messages when it attempts to respond to the original spoofed packet, or that other hosts who have been spoofed that have no TCP installed, complain.

CIDR	IP Block	b16	NHI
6/7	0.0.7.208/29	0x000007d0	6:2(3) 6:7(0) 6:8(0)
...			
33/2	0.0.10.52/30	0x00000a34	33:0(1) 33:1(2)
33/3	0.0.10.48/30	0x00000a30	33:1(1) 33:2(0)
33/4	0.0.10.44/30	0x00000a2c	33:2(1) 33:3(3)
33/5	0.0.10.40/30	0x00000a28	33:1(0) 33:3(0)
33/6	0.0.10.16/29	0x00000a10	33:2(2) 33:8(0)
33:9(0)			
33/7	0.0.10.8/29	0x00000a08	33:3(2) 33:10(0)
33:11(0)			
33/8	0.0.10.0/29	0x00000a00	33:3(1) 33:12(0) 33:13(0)

```

--- Phase III: add static routes
## Net config: 741 routers and hosts
## Elapsed time: 3.185 seconds
** Running for 5678900000000000 clock ticks (== 567890.0 seconds sim
time)
Spoofing IP -1->60
Host 33:13 PKT RCVD from 60 ID: 13130 marked by: |6:2| AS:7 SID:2
LINK:3
** Bad header: LAN dropping SSF.OS.PacketEvent@55bb93
Spoofing IP 60->16
Host 33:13 PKT RCVD from 16 ID: 13132 marked by: |6:2| AS:7 SID:2
LINK:3
** Bad header: LAN dropping SSF.OS.PacketEvent@8de462
Host 33:13 PKT RCVD from 5172 ID: 13137 marked by: |33:1| AS:-1 SID:1
LINK:2
208.246812205 host 33:13 in ip: SSF.OS.ProtocolException: in {Host
33:13 {:[NIC 33:13(0) 0.0.10.3/29 :: bitrate=1.0E7, latency=0.0010,
buffer= 9223372036854775807]}}: protocol icmp has no registered
implementation

```

Looking at the above, we can see that our spoofing host changes the source IP address for a packet being sent to 33:13 to 0.0.0.60. The packet is received by 33:13 with the false IP address being noted, but also containing *traceback* information: Network 6, SID 2, link 3. When we look this up in our routing table above we notice that it came from either host 6:7 or 6:8. The reason we cannot be certain which host truly sent the packet is because they are both on a shared network segment. Unfortunately, our approach does not allow a *traceback* further than this because it cannot distinguish packets coming from two different sources over the same link. Authentication is our solution around this, and the next 2 simulations deal with this.

4.4 Traceback in Mixed Transit-leaf Segments

In this simulation, we now begin to investigate the use of authentication as a means of distinguishing packets that have already been marked by downstream *traceback* enabled routers, and those that have not been or packets that contain bogus *traceback* information; both would need to be marked regardless.

To get right to the point in this simulation, we create a mixed transit-leaf segment and then use the leaf node of that segment as our spoofing device (Host 22). This topology is extended from prior client networks as depicted below in figure 5. We behaviorally simulate IPSec in this current run, because of limitations with SSFnet, by creating an extra field in our *traceback* header called ‘*authenticated*’. If a packet with *traceback* information is received, but it has not been authenticated (realistically, from use of IPSec), then we mark it normally and output a warning in the simulation. In the real world this information could certainly be used for intrusion detection.

In our first run, our spoofing host will send false *traceback* information and spoofed source IP (identically mimicking *traceback* information as if Host 23 had sent it), along with its payload, out on interface 0. In the second run, we send a packet from host 23 normally and show that all proceeds as planned.

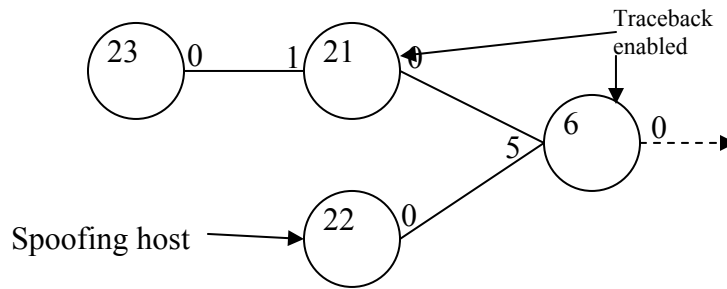


Figure 5 – A mixed mode segment

```

CIDR          IP Block          b16          NHI
--           0.0.0.0/20      0x00000000
...
6/13         0.0.7.128/29      0x00000780   6:6(5) 6:21(0) 6:22(0)
6/14         0.0.7.192/30      0x000007c0   6:21(1) 6:23(0)
...
29/8         0.0.10.128/29     0x00000a80   29:3(1) 29:12(0) 29:13(0)
NHI Addr          CIDR Level          IP Address Block  % util
**                Using specified 1.0ns clock resolution
  
```

```

--- Phase III: add static routes
## Net config: 703 routers and hosts
## Elapsed time: 1.462 seconds
** Running for 5679000000000000 clock ticks (== 567900.0 seconds sim
time)
Unauthenticated Header detected!!
6:23 packet from
(IPv4/tcp:s=0.0.7.194:d=0.0.10.131:n=0.0.0.0:tos=0:ttl=128) sp=10001
dp=10
    previous: ID: 718 marked by: |6:21| AS:-1 SID:21 LINK:1
Host 29:13 PKT RCVD ID: 719 marked by: |6:6| AS:7 SID:6 LINK:5
    new: ID: 719 marked by: |6:6| AS:-1 SID:6 LINK:5
-----run(2) with no spoofing-----
-----
--- Phase III: add static routes
## Net config: 703 routers and hosts
## Elapsed time: 1.472 seconds
** Running for 5679000000000000 clock ticks (== 567900.0 seconds sim
time)
6:23 packet from
(IPv4/tcp:s=0.0.7.194:d=0.0.10.131:n=0.0.0.0:tos=0:ttl=128) sp=10001
dp=10
Host 29:13 PKT RCVD ID: 718 marked by: |6:21| AS:7 SID:21 LINK:1
202.564007951 host 29:13 in ip: SSF.OS.ProtocolException: in {Host
29:13 {:[NIC 29:13(0) 0.0.10.131/29 :: bitrate=1.0E7, latency=0.0010,
buffer= 9223372036854775807]}}: protocol icmp has no registered
implementation

```

As can be seen in run 1, a bogus *traceback* header was detected. The previous marking shows that it had been spoofed as having come from host 23 (SID 21, interface 1), which can be verified in run 2. The true origin of the spoofed packet is tracked to switch 6, link 5. Again though, because host 22 shares a link with router 21, we cannot be certain if router 21 generated the packet or if it was from our spoofing host (without link-to-link authentication).

4.5 Traceback on a Pure Transit Segment

In this final simulation we test our authentication mechanism and its ability to successfully distinguish unmarked or falsely marked packets, from those legitimately marked with *traceback* information in a pure transit segment. In this scenario, the attacker attempts to spoof it's the

packet as if it were coming from Host 23 having been marked by some router between host 23 and host 6. Host 23 happily sends packets as usual and router 21 is configured for non-*traceback*. The new topology is as 4.4, with modifications as per figure 6.

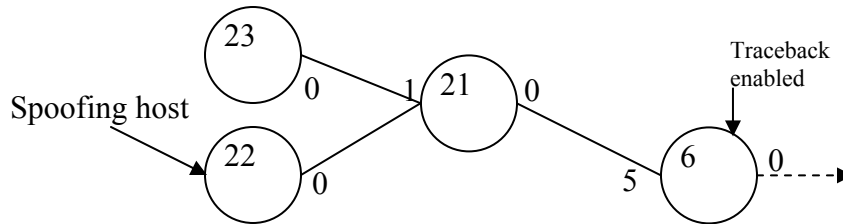


Figure 6 – A Pure Transit Segment

```

CIDR          IP Block          b16          NHI
--           0.0.0.0/20          0x00000000
...
6            0.0.7.128/25          0x00000780    6
...
6/5          0.0.7.196/30          0x000007c4    6:5(2) 6:6(0)
...
6/13         0.0.7.128/29          0x00000780    6:6(5) 6:21(0)
6/14         0.0.7.192/30          0x000007c0    6:21(1) 6:22(0) 6:23(0)
...
29/8         0.0.10.128/29          0x00000a80    29:3(1) 29:12(0) 29:13(0)
NHI Addr          CIDR Level          IP Address Block    % util

**              Using specified 1.0ns clock resolution

--- Phase III: add static routes
## Net config: 703 routers and hosts
## Elapsed time: 1.788 seconds
** Running for 5678900000000000 clock ticks (== 56789.0 seconds sim time)
6:23 packet from
(IPv4/tcp:s=0.0.7.131:d=0.0.10.131:n=0.0.0.0:tos=0:t1=128) sp=10001
dp=10
Host 29:13 PKT RCVD from 0.0.7.131 ID: 1632 marked by: |6:6| AS:7 SID:6
LINK:5
6:23 packet from ID: 1633 marked by: |6:6| AS:-1 SID:6 LINK:0 sp=10
dp=10001
Unauthenticated Header detected!!
previous: ID: 1634 marked by: |6:21| AS:-1 SID:21 LINK:1
new: ID: 1635 marked by: |6:6| AS:-1 SID:6 LINK:5
  
```

In the above output we see that host 29:13 received packet number 1632 from IP 0.0.7.131 and that no prior bogus *traceback* headers were detected. We then see that packet id 1634 was found to be bogus – claiming to have come from SID 21, interface 1.

However this could not have been possible since we have intentionally configured Router ID 21 to be a non-traceback router. This unauthenticated header is detected and is marked by Router ID 6 on interface 5; the first *traceback* enabled router the packet comes across.

4.6 Limitations

This approach does have its limits. Immediately clear is that it cannot determine which host on a shared link produced the packet. If all nodes on that link are truly end hosts then, by our assumption that all hosts are to be untrusted, we are precluded from using an authentication mechanism to distinguish them. We see that this was the case in experiment 4.4 and essentially in 4.5 (since packets were aggregated at Router ID 21 and sent out on link 0).

5.0 IDEAS FOR THE FUTURE AND CONCLUSIONS

5.1 Future work.

Ultimately, our simulation is limited by what protocols SSFnet supports, and its implementation subtleties. Currently, SSFnet does not support IPsec or any form of tunneling (directly). Future work would try to include that along with some analysis for actually folding traceback information into IPsec.

Further, the jury is still out on the issue of performance overhead. Ideally, our future work would focus on this very issue. Specifically, how much bandwidth drop do we see when we have different levels of *traceback* deployment and use of IPsec with authentication headers and encryption? Can *traceback* enabled switches keep packets on the fast-path by using the TOS field described earlier or is there some unforeseen issue with this? If indeed this deterministic approach were found to be unacceptable in terms of performance, experimenting with probabilistic approaches (and high marking probabilities) along with our marking scheme would also provide some further work.

Some things would be difficult to ascertain in regards to hardware implementation. In hardware routing/switching devices, plenty of tricks can be implemented to perform tasks in parallel. Given the non-deterministic nature of application level software like SSFnet (particularly in Java), such things as multithreading for simulating parallel processing would likely not be possible.

Another area that would be useful to look into would be automated response to an attack using our traceback approach. In a DoS attack we could conceivably use this mechanism to request rejection of the offending packets from routers deep in the core of the internet – far enough away from the victim host to not consume any of its bandwidth combating the attack.

Finally, the vast aggregation of traffic that occurs at a true AS boundary would make tracking a single switch challenging. Exploring the possibility of actually tunneling the *traceback-encoded* packets, from the ingress *traceback* enabled switch, to a compliant BGP router could alleviate many possible complications (packets could then stay on the fast-path to the BGP router, and once de-capsulated, put into stream normally)

5.2 Conclusions

This approach addresses deficiencies found in previous approaches in many ways. First, if we can traceback with accuracy to a single host then we can traceback a flood of hosts that would be signature of a DDoS attack and therefore we have lost nothing in the fight against DDoS. Further, none of the approaches discussed thus far deals with the fact that NAT is becoming nearly ubiquitous – almost assuring us that IP source addresses inside the NATed network will not be unique. Lastly, our hierarchical approach allows an entire corporate LAN to be traceback enabled by deploying and configuring a single aggregate switch (at the expense of traceback granularity), and can readily identify both the network (by AS id) and the host (by SID:LINK).

This problem of granularity is also realized at the AS level – whose scope generally cannot be influenced by anyone outside the ISP (the ISP may not even have complete control). The problem with granularity at the AS level is that there is a huge gap between identifying the AS and identifying a single switch within that AS. Tracking down that single switch could become the equivalent to tracking down a needle in a haystack. The use of a MAC address as a SID may help by identifying the owner of the router/switch or, possibly, an appropriate router anywhere inside the AS could mark the packet again. This has its limitations though and also brings up another valid observation.

The process of validating an attack origin could become tedious on an Internet scale. On the LAN scale this could be done quite painlessly given that the attack origin is already heavily constricted and administrative procedures to obtain such information would (hopefully) be less. On the Internet there may actually be *unwillingness* to co-operate on behalf of the entity that owns the attacking host. Such co-operation could be part of a socially engineered attack or could end up with unwanted legal ramifications for the owner of the attacking host.

While IP-level traceback can be an invaluable tool toward tracking down an *attacking host*, it cannot identify the *attacker* and it is by no means complete. However, this traceback information can be used to check logs and look for other forensic information that could lead to the identity of the attacker. A more permanent solution would necessitate legal disclosure of such information and action that would result in accountability, as well as authentication of IP packets as is possible with IPv6¹⁸

As with any approach there are drawbacks. First, writing to the options field removes the ability of an end-host to make use of the options field at all. Given current statistics on the usage frequency of the IP options field¹⁹ it is unlikely this will be a problem. It is precisely because of this low-usage status that brings us to another drawback

Adding anything to the options field will force every non-iptraceback compliant router along the path to suffer the overhead of having to bring the packet to the software layer in order to evaluate the options field which has been known to cause DoS when used en masse[11].

To date, none of the approaches to IP *traceback* propose a course of action or even a means of combating an attack (DDoS or otherwise). The approach supplied in this paper would allow a victim to send a message to an authority on the attacking AS (which would be known from the *traceback* packet) to block any *traceback* packet with the SID of the attacking switch (which would also be known). Alternatively, an intermediate router could block packets from the origin AS if a router is compromised inside the origin AS. However, a call to arms such as this must be done with surpassing care and with a method of authentication; lest it be used as a wretched device for DoS.

In this work we show that improved deterministic traceback can be efficiently and robustly used to traceback packets to their source, incrementally deployed in a LAN environment and is completely transparent to the rest of the network and host for which it

marks (and is thus backward compatible), and that it is also robust against interference when used in conjunction with a packet authentication mechanism.

References

-
- ¹ Practical Network Support for IP Traceback, Stefan Savage, David Wetherall, Anna Karlin, Tom Anderson SIGCOMM 2000
 - ² Advanced and Authenticated Marking Schemes for IP Traceback, Dawn X. Song, Adrian Perrig Proceedings IEEE Infocomm 2001
 - ³ IP Traceback With Deterministic Packet Marking, Andrey Belenky and Nirwan Ansari IEEE Communication Letters April 03
 - ⁴ <http://www.iitg.ac.in/studs/Members/ryanachu/acad/acad/ppts/traceback.ppt>
 - ⁵ Security fun with OCxmon and cflowd, Sager, Glenn; Internet 2 Measurement Working Group
 - ⁶ Hash-Based IP Traceback, Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones
 - ⁷ A Layer-2 Extension to Hash-Based IP Traceback, Hiroaki HAZEYAMAy, Masafumi OEyy, and Youki KADOBAYASHIy,
 - ⁸ ICMP Traceback Messages, Steven Bellovin AT&T Labs Research, March 2000
 - ⁹ Centertrack: An IP overlay network for tracking DoS floods, Robert Stone, In Proceedings of the USENIX Security Symposium
 - ¹⁰ Tracing anonymous packets to their approximate source, Hal Burch and Bill Cheswick. In Proceedings of the USENIX Large Installation Systems Administration Conference, Decemeber 2000. USENIX.
 - ¹¹ On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack (2000), Kihong Park, Heejo Lee
 - ¹² Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing, P. Ferguson and D. Senie. RFC2267, IETF, January 1998.
 - ¹³ 2002 CSI/FBI Computer Crime and Security Survey. Computer Security Issues and Trends Vol. No. 1 Spring 2002
 - ¹⁴ Rekhter & Li RFC 1771
 - ¹⁵ Watson, Paul: Slipping in the Window: TCP Reset attacks
 - ¹⁶ <http://www.bgpexpert.com/>
 - ¹⁷ Mogul & Deering RFC 1191
 - ¹⁸ S. Deering & R. Hinden. RFC 2460
 - ¹⁹ Cisco IP Options Selective Drop
http://www.cisco.com/en/US/products/sw/iosswrel/ps1829/products_feature_guide09186a0080104376.html