Network Traceback

Eric Stone

DoS Attacks

- Easy to launch
- Hard to trace
- Zombie machines
- Fake header info

The Ultimate Goal

- Stopping attacks at the source
- To stop an attack at its source, you need to know where it is coming from
  - This is where traceback comes in

The Ultimate Goal

- The traceback problem
  - Finding the origin of the attack
- Legal or technical remedy still needed once attack source is found

Overview of Traceback Ideas

- All involve changing how routers operate
- Traceback methods need to be:
  - backward compatible
  - efficient
- Three methods
  - Two ideas for traceback of DoS attacks
  - One for traceback of any packet

Network Support for IP Traceback

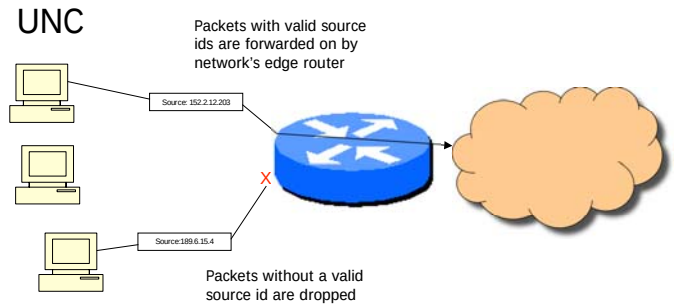
Ideas that don't work

- Ingress filtering
  - Stopping attack packets from entering the network
- Link testing
  - Querying upstream router for an attacks source
  - Input Debugging, Controlled Flooding
- Logging
  - Logging all network traffic
- ICMP Traceback
  - Router sends route messages to a packets destination

Ingress Filtering

- Block forged source addresses at router
- Can prevent attacks
- Must be deployed at network edge
- Requires almost universal deployment
- Addresses can still be forged in the valid range
  - But in this case the network hosting the attack is identified

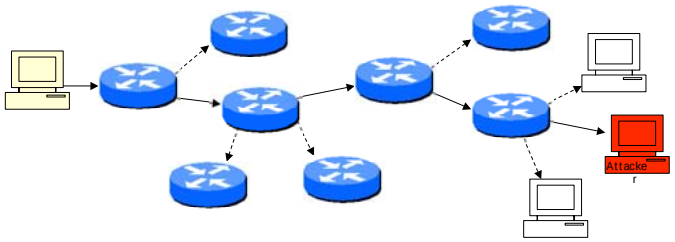
Ingress Filtering



Link Testing - Querying

- Recursively determining which link a packet came from
- Traceback must occur during attack
- Must be supported by router OS

Link Testing - Querying



Link Testing - Querying

- **Computationally Intensive**
- **Requires use of an attack signature**
- **Upstream IPSs must be willing and able to support this**

Link Testing – Controlled Flooding

- **Operates similarly to querying but uses flooding of upstream routers to determine which ones the attack passes through**
- **a DoS attack in itself**
- **Requires lots of network resources**
- **Problematic to use on attacks from multiple sources**
- **Require reliable network map**

Logging

- Keep logs of traffic flow at the routers
- Very resource intensive, especially on high bandwidth links
- Sharing of logs a practical problem

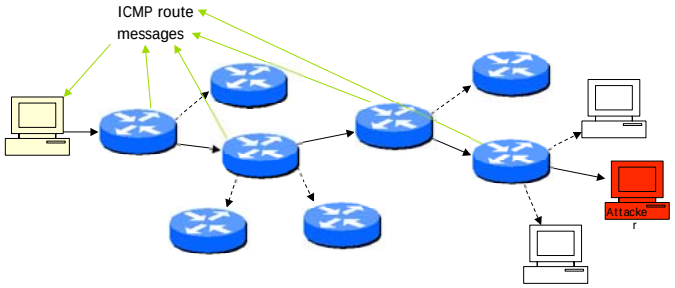
ICMP Traceback

- Router forwards an ICMP packet with route info to packet destination
- Done for only a small sampling of packets
- Enough ICMP messages should be generated by a large DoS attack to enable the victim to be able to reconstruct the attack route
- ICMP packet may be filtered during an attack

ICMP Traceback

- Requires router to be able to report on a packet's source (input debugging capacity)
- Every router on attack route must participate
- Messages must be signed to prevent attacker sending false messages, requiring a key distribution infrastructure
- Authors feel idea is promising especially if combined with their own proposals

ICMP Traceback



Traceback by Marking

- Use an algorithm where the routers will probabilistically mark packets with route information
- Route will be reconstructed from the marked packets that arrive at the victim

Traceback by Marking

- Mentioned by Burch and Cheswick
- Doesn't require interactive cooperation between ISPs
- Can be used after attack is finished

Assumptions

- Attackers may try to fool system
- Attacks may involve many sources
- Routers largely uncompromised
- Route between attacker and victim is relatively stable
- Attack must consist of a large number of packets

Marking Algorithms

- Node Append
- Node Sampling
- Edge Sampling

Node Append

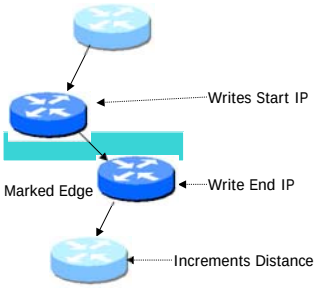
- A list of all routers traversed is stored in the packet
- Easy to reconstruct
- Lots of router overhead
- No way to guarantee there will be enough space reserved in the packet
- Easy to fool by attacker
  - Attacker can fill up space reserved for the list
  - Attacker can add false information to end of list

Node Sampling

- A 'node' field is reserved in the packet header
- At each router the IP of the router is written into the node field with probability p (p>0.5)
- Slow to reconstruct
  - A large number of packets must be received before more distant routers can reliably be identified
- Not robust against multiple attackers
  - Overlapping routes at the same distance may be impossible to distinguish

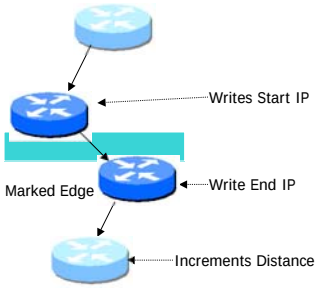
Edge Sampling

- Instead of nodes, encode the edges of segments of the route
- Requires three new fields be reserved in the packet header, start and end IP and distance



Edge Sampling

- Each router writes its IP in the start field and sets distance to 0 with probability p (p = 1/25)
- All other routers increment the distance field and if that field is 0 then they write their own IP in the end field



Edge Sampling

- This creates a collection of edges stored in packets arriving at the victim
- Robust against multiple attackers
- Attackers can't forge edge between themselves and the victim
- Allows for non-participating routers in the path
- Difficult to convince people to modify the packet headers and then update infrastructure to support it

Compressed Edge Fragment Sampling

- Modification of the edge sampling method to reduce space requirements and make it compatible with the current IP header

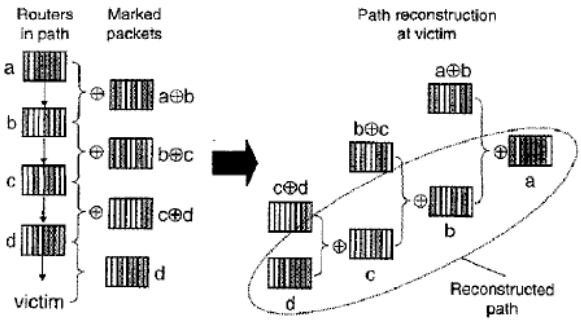
XOR Encoding

- First router will mark a packet with its IP
- Next router will XOR its IP with the one from the previous router, creating an edge-id

XOR Encoding

- Last hop before victim will only have one IP address
- Because b XOR a XOR b = a, victim can reconstruct all of the edges in the attack starting from the un-encoded IP
- This reduces IP storage space needed by half
  - Only one 32 bit IP field is needed instead of two

XOR Encoding



Edge-id Fragmentation

- Edge-ids are divided into non-overlapping fragments
- One of these fragments is stored is randomly stored along with its offset

Edge-id Fragmentation

- This further reduces space requirements
- Requires more packets arrive at victim in order to reconstruct route

Edge-id Fragmentation

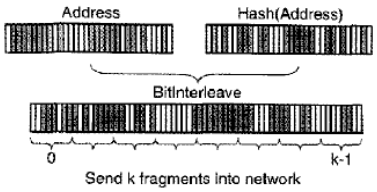
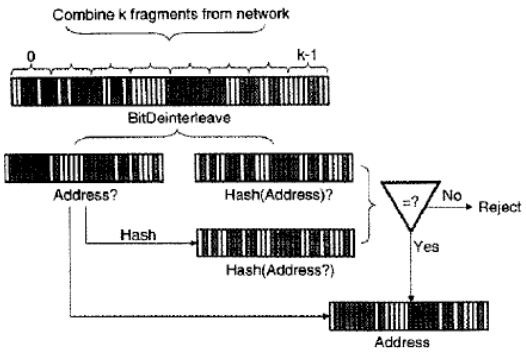


Fig. 6. Each router calculates a uniform hash of its IP address once, at startup, using a well-known function. This hash is interleaved with the original IP address (the original address on odd bits, the hash on even bits). The resulting quantity is then broken into k fragments, which the router selects among randomly when marking a packet. Although it is not shown, each of these fragments is further labeled with its offset. The next downstream router uses this offset to select the appropriate fragment to XOR—thereby encoding part of an edge.

Edge-id Fragmentation Hashing

- With multiple attackers, there is a possibility that fragments at the same distance will be ambiguous
- To decrease the chances of this being a problem, each address is interleaved with a hash of the address at the router
- Reconstructed addresses must then match the reconstructed hash in order to be accepted
- This doubles the length of the value that must be stored (as fragments) and increases the number of packets the victim must receive to reconstruct the route

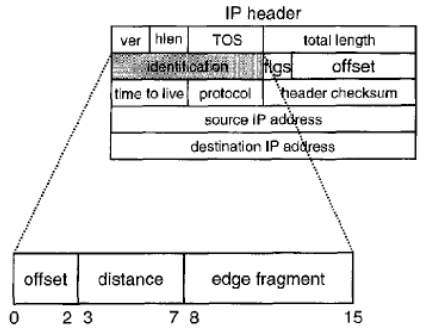
Edge-id Fragmentation Hashing



IP Header Encoding

- The authors suggest that their method could be implemented by using the 16 bit identification field to store the traceback data
- This field is normally used for numbering fragmented packets, which make up less than 0.25% of internet traffic

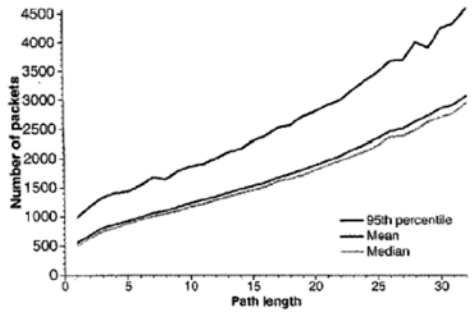
IP Header Encoding



IP Header Encoding

- Legacy support could be maintained by sending traceback data in a ICMP echo reply packet for already fragmented packets and by setting the do not fragment flag on un-fragmented packets that are marked
- Most experts suggest that fragmentation should be avoided anyway, so the authors feel using overloading the identification field is not bad style

Packets Needed to Reconstruct Route



However….

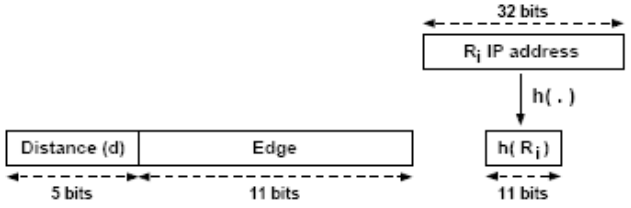
- This method requires high processing overhead by the victim and produces a large number of incorrect routes or false positives
- Or so claim the authors of the next paper considered

Advanced and Authenticated Marking Schemes for IP Traceback

Advanced Marking Scheme I

- If the victim has a map of upstream routers, it don't need to know every router's full IP address to reconstruct a path
- Encode router's IP address using hash functions
  - Each router has its own, publicly known hash function
- Store this information in the header's ID field as before

Advanced Marking Scheme I

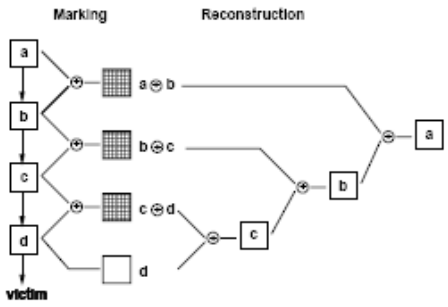


- **11 bit hashes, leaving enough space to encode 32 hops in header ID field**

Advanced Marking Scheme I

- Decode as before, a XOR b XOR a = b
- Encodes order of upstream routers using hashes
- Victims can compute a router's hash value from an upstream network map

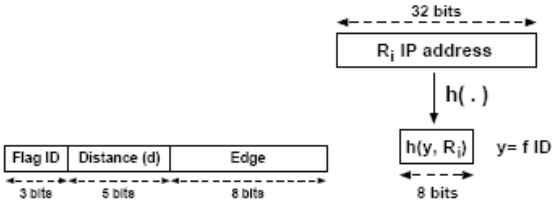
Advanced Marking Scheme I



Advanced Marking Scheme II

- Same as Advanced Marking Scheme I except instead of using two hash functions use 2^a hash functions derived from a series of hash functions ($h_i(x) = g(<i, x>)$)
- This requires shortening the hash output to leave space for a *w* bit flag id to indicate what *i* was used by the hash

Advanced Marking Scheme II



- IP address is hashed into a 8 bit value
- Flag ID field used to identify hash

Advanced Marking Scheme II

- Decoding this scheme is similar to before but makes use of a threshold value *m*
- Based on this threshold, an upstream router is only considered part of the attack path if more than *m* of the 2w possible hash variations arrive at the victim
- Reduces likelihood of false positives

Advanced Marking Scheme II

- The Advanced Marking Scheme is intended to remedy the shortfalls of the Fragment Marking System by:
  - Producing fewer false positives
  - Requiring fewer packets to reconstruct a route

Advanced Marking Scheme II Simulation

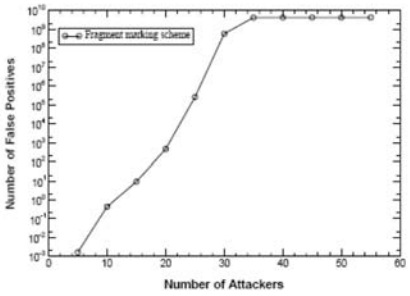


Fig. 10. False Positives for FMS

- High number of false positives produces by FMS when faced with multiple attackers

Advanced Marking Scheme II Simulation

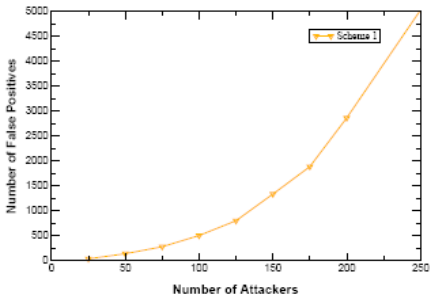


Fig. 7. False Positives for Advanced Marking Scheme I

- Reduced number of false positives seen using AMS I

Advanced Marking Scheme II Simulation

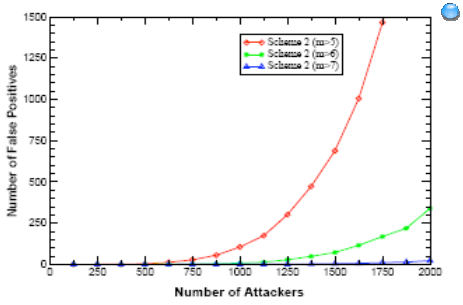


Fig. 8. False Positives for Advanced Marking Scheme II

- Even fewer false positives seen when using AMS II
  - Requiring 6 or more hash variations seen before accepting a router at part of attack route

Advanced Marking Scheme II Simulation

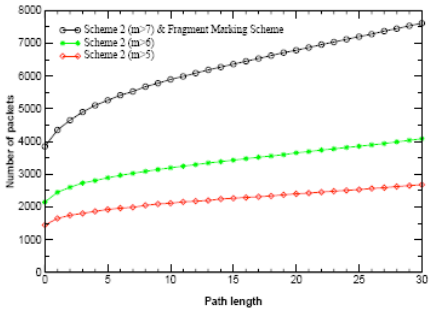


Fig. 13. Number of Packets Required for Reconstruction ($q = 1\%$)

- Packets required for route reconstruction with marking probability set at 1 percent

Advanced Marking Scheme II Simulation

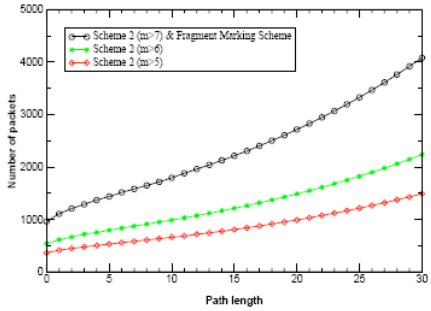


Fig. 12. Number of Packets Required for Reconstruction ($q = 4\%$)

- Packets required for route reconstruction with marking probability set at 4 percent

Authenticated Marking Scheme

- One problem with all of these schemes is that a compromised router on the attack path could change the packet markings to create a false path and disguise the real path
- Digitally signing the packet markings is expensive both in terms of space and computation

Authentication with a MAC

- Marked packets can be authenticated using a Message Authentication Code (MAC)
  - The MAC a secret key shared between a marking router and victim
- By using this key in the hash, each router produces a unique marking that cannot be forged by a compromised router
- For this to work some secure method of key exchange is needed

Using Time-Released Chains

- The problem of key exchange can be overcome by each router using a hash chain of MAC keys
- Each of these keys is associated with a time interval and packets marked during that interval are marked using that interval's key
- After a long enough time for all the packets marked during an interval to have arrived, the MAC key for that interval is publicly released, allowing attack victims trying to reconstruct an attack path to authenticate packets marked by that router

Packet Marking Schemes

- Could be incorporated into most existing infrastructure with only changes to the routers' OS
- Only effective again attacking involving large numbers of packets (DoS attacks)

Single Packet IP Traceback

Source Path Identification Engine

- A method for the traceback of a single packet
- Useful against non-DoS attacks (Ping of Death, LAND) or individualized attacks
- Must work against hostile opponents and networks
- Must respect user privacy
- Must not require to many system resources (storage, processor)
- Must be able to trace packets that undergo transformations (encapsulation, generation or duplication)

Source Path Identification Engine

- A signature is stored in a packet digest for each packet that passes through a router
- These digests cover a particular traffic flow in a router and cover a specific time interval
- A recursive lookup of an attack packet's signature will reveal its route

Digest Input

- Packets are uniquely identified by the first 24 invariant bytes of the packet
- 16 bytes of the header, excluding the TOS, TTL, checksum and options
- First 8 bytes of the payload

Digest Input

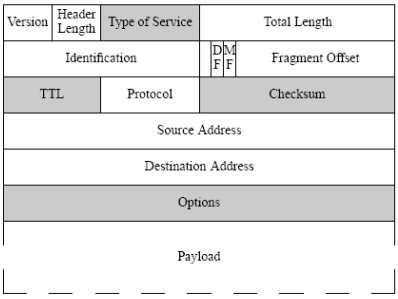
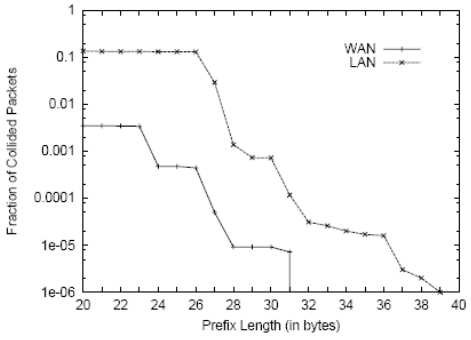


Fig. 1. The fields of an IP packet. Fields in gray are masked out before digesting, including the Type of Service, Time to Live (TTL), IP checksum, and IP options fields.

Digest Input

- Using these fields to uniquely identify a packet resulted in collision rates of:
  - ~ 0.00092% in WAN tests
  - ~ 0.139% in LAN tests
- The LAN had a smaller address range and more packet duplication

Digest Input



Bloom Filters

- Packet hashes are stored in Bloom Filters
- Bloom filters hash the packet using *k* hash functions that produce outputs of *n* bits and store the results in a 2n bit sized digest
- The digest is initially set to 0 and then bits are set to 1 as packets are hashed to those locations
- These digests are stored for a finite period of time and then overwritten

Bloom Filters

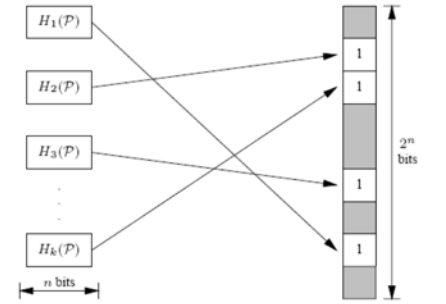


Fig. 3. For each packet received, SPIE computes k independent n-bit digests, and sets the corresponding bits in the 2^n-bit digest table.

Bloom Filter Hash Functions

- The hash functions must have a uniform distribution over their result space, i.e. be good hash functions
- Collisions in one hash function must be independent of collisions in another, i.e. be universal hash families
- They must be quick to compute

Source Path Isolation Engine

- The authors outline a SPIE infrastructure which implements this system of Bloom Filter digests across an ISP's network
- Data generation agents at the routers produce the hash filters for that router
- These are collected SPIE Collection and Reduction Agents, which have the role of performing traceback for their region of the network
- All activity is controlled by a SPIE Traceback Manager

Source Path Isolation Engine

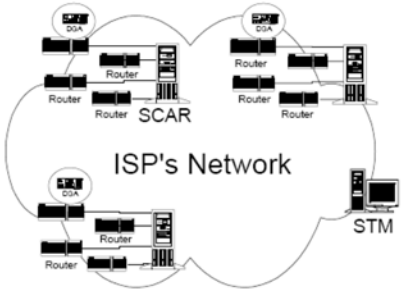


Fig. 4. The SPIE network infrastructure, consisting of Data Generation Agents (DGAs), SPIE Collection and Reduction Agents (SCARs), and a SPIE Traceback Manager (STM).

Source Path Isolation Engine

- Requests for traceback include the offending packet, the point it exited the local SPIE's domain and the time of the packet's arrival at the exit point
- The SPIE then checks the appropriate digests and returns an attack graph that either indicates the host that the packet originated from or where it entered the SPIE's network

Source Path Isolation Engine

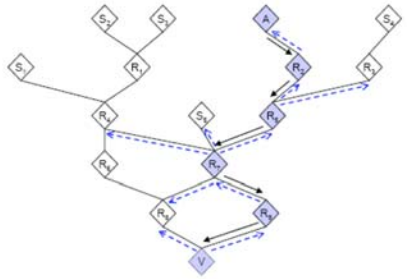


Fig. 6. Reverse path flooding, starting at the victim's router, V, and proceeding backwards toward the attacker, A. Solid arrows represent the attack path; dashed arrows are SPIE queries. Queries are dropped by routers that did not forward the packet in question.

Transformation Processing

- The system must be able to trace packets that undergo transformations in route
- The SPIE must store enough information to be able to recover any changes that occurred to the fields it hashes into the digest
- These can include: fragmentation, network address translation, ICMP messages, IP-in-IP tunneling and IP security

Transformation Processing

- The SPIE maintains a transformation lookup table along with each digest it stores
- The TLT stores 29 bits of the digest, the type of transformation and any irrecoverable data, either in the 32 bit packet data section or in an external data structure
- The rarity of transformations allows for the partial digest storage

Transformation Processing

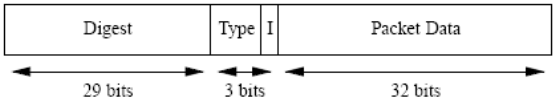


Fig. 5. A Transform Lookup Table (TLT) stores sufficient information to invert packet transformations at SPIE routers. The table is indexed by packet digest, specifies the type of transformation, and stores any irrecoverable packet data.

Transformation Processing

- The SPIE maintains a transformation lookup table along with each digest it stores
- The TLT stores 29 bits of the digest, the type of transformation and any irrecoverable data, either in the 32 bit packet data section or in an external data structure
- The rarity of transformations allows for the partial digest storage
- Packets not found in the digest are then checked against the TLT

Source Path Identification Engine

- Allows for the origin of individual packets to be traced
- Has time constraints that packet marking schemes do not
- Would require a much larger investment in infrastructure changes than packet marking

Conclusions

- Both schemes have their advantages
- No real world implementation of either
- Implementation of Fragment Marking seems more likely, but not without the enthusiastic support of a major vendor and/or major ISPs

References

Network Support for IP Traceback
Stefan Savage, David Wetherall, *Member, IEEE,* Anna Karlin, and Tom Anderson

Advanced and Authenticated Marking Schemes for IP Traceback
Dawn Xiaodong Song and Adrian Perrig
fdawnsong, perrigg@cs.berkeley.edu
Computer Science Department
University of California, Berkeley

Single-Packet IP Traceback
Alex C. Snoeren, *Student Member, IEEE*, Craig Partridge, *Fellow, IEEE*,
Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, *Member, IEEE*,
Beverly Schwartz,
Stephen T. Kent, and W. Timothy Strayer, *Senior Member, IEEE*