Traceback of Single IP Packets Using SPIE

W. Timothy Strayer, Christine E. Jones, Fabrice Tchakountio, Alex C. Snoeren,
Beverly Schwartz, Robert C. Clements, Matthew Condell, and Craig Partridge

BBN Technologies
10 Moulton Street, Cambridge, MA 02138

{strayer, cej, ftchakou, snoeren, bschwart, clements, mcondell, craig}@bbn.com

ABSTRACT

The design of the IP protocol makes it difficult to reliably identify the originator of an IP packet. IP traceback techniques have been developed to determine the source of large packet flows, but, to date, no system has been presented to track individual packets in an efficient, scalable fashion. We present SPIE, the Source Path Isolation Engine, a hash-based technique for IP traceback that generates audit trails for traffic within the network, and can trace the origin of a single IP packet delivered by the network in the recent past.

1 Introduction

Attack and other malicious network traffic has serious financial and national security consequences, and has led to the research and development of many types of defense mechanisms. One such mechanism is a system for tracing packets back to their points of origin. Such IP traceback systems are important first steps in making attackers (or, at least, the systems they use) accountable. Yet, there are a number of significant challenges in the construction of such a tracing system including determining which packets to trace, maintaining privacy, and minimizing cost, both in time spent tracking packets and in storage used to keep information.

BBN Technologies has developed SPIE, the Source Path Isolation Engine, a single-packet IP traceback system that provides the ability to identify the source of a particular IP packet given a copy of the packet to be traced, its destination, and an approximate time of receipt. Historically, tracing individual packets has required prohibitive amounts of memory; one of SPIE's key innovations is to reduce the memory requirement (down to 0.5% of link bandwidth per unit time) through the use of Bloom filters [2]. By storing only packet digests, and not the packets themselves, SPIE also does not increase a network's vulnerability to eavesdropping. SPIE therefore allows routers to efficiently determine if they forwarded a particular packet within a specified time interval while maintaining the privacy of unrelated traffic. SPIE also traces packets across *transforms*, where the packet is significantly changed within a router as part of the forwarding process. Such transforms include fragmentation, source routing, tunneling, and reflections like ICMP query and error packets.

This work was sponsored by the Defense Advanced Research Projects Agency (DARPA) under contract No. N660001-00-C-8038. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied.

2 Current Traceback Systems

Currently proposed IP traceback systems employ one of three basic methods: route inference based on traffic flows, end-host supported packet auditing, and network supported packet logging. Each method of traceback requires a different balance of implicit and explicit support to enable traceback of packet transformations.

Route inference, effective only in tracing large packet flows, uses controlled flooding of network links to measure flow variations and infer a flow's path [3]. Traceback is based on packet flow characteristics rather than individual packet characteristics. The form of a packet does not affect traceback, just the rate of packets.

Individual packet characteristics are important within traceback systems that actively audit packets. Packet auditing based on end-host support requires that intermediate routers on a packet's path notify the destination host that it forwarded the packet. The destination host collects these notifications to construct the traversed path. Trace information can be delivered to the end host within the packet itself [13, 16] or within a separately routed packet [1].

In the third traceback method, the network rather than end hosts is tasked with maintaining audit trails of network traffic. Such traceback systems log a representative amount of packet content at the routers. This content can range from the entire packet [11] to a 32-bit digest [14, 15]. Extraction techniques are used to access the packet logs and perform traceback.

3 Overview of SPIE

SPIE falls into the third traceback method. It is a log-based traceback system that uses auditing techniques at network routers to support the traceback of individual IP packets. Traffic auditing is accomplished by computing and storing 32-bit packet digests rather than storing the packets themselves. Every packet traversing a SPIE-enhanced router is recorded in a *digest table*; digest tables are paged at a specified rate and are representative of the traffic forwarded by the router during a particular time interval. A cache of digest tables is maintained for recently forwarded traffic.

If a packet is determined to be offensive by some intrusion detection system (or judged interesting by some other metric), a query is dispatched to SPIE which in turn queries routers for packet digests of the relevant time periods. SPIE then builds a graph of the routers visited by the packet.

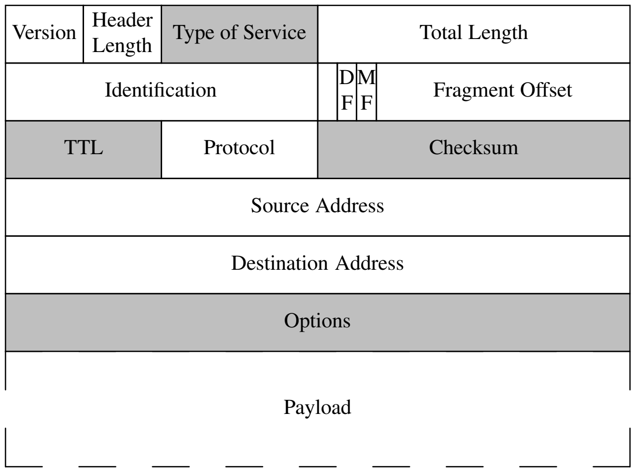

Figure 1: The fields of an IP packet. Fields in gray are masked out before digesting, including the Type of Service, Time to Live (TTL), IP checksum, and IP options fields.

3.1 Packet Digesting

Three somewhat disparate requirements were considered in determining the optimal digest input.

1. *Unique packet representation*: To decrease digest collisions, the digest input must uniquely represent every packet. Two different packets should not produce identical digest input.
2. *Identical digest input*: The digest input of a packet must be identical at all hops along the forwarding path to produce a trail of identical digests for traceback.
3. *Limited digest input size*: It is desirable to limit the size of the digest input both for performance and reasons concerning transforms.

In their work on trajectory sampling, Duffield and Grossglauser encountered similar requirements while sampling a subset of forwarded packets in an attempt to measure traffic flows [4]. SPIE uses the same approach of masking variant packet content and selecting an appropriate-length prefix of the packet to input to the digesting function. The choice of invariant fields and prefix length is slightly different, however.

Figure 1 shows an IP packet and the fields included by the SPIE digesting function. SPIE computes digests over the invariant portion of the IP header and the first 8 bytes of the payload. The mutable header fields (i.e., TTL and checksum) are masked prior to digesting, as well as the TOS field that may be frequently modified by packet marking protocols. Also masked are the IP options because options often cause routers to rewrite the option field at various intervals. To ensure that a packet appears identical at all steps along its route, SPIE masks or compensates for these fields when computing the packet digests. Packet transformation may occasionally modify the fields used in the digest function, so SPIE must handle these situations.

Study was required to determine that the first 8 bytes of payload, along with the header as masked in figure 1, are sufficient to differentiate almost all non-identical packets. Figure 2 presents the rate of packet collisions for an increasing prefix length for two representative traces: a WAN trace from an OC-3 gateway router, and a LAN trace from an active 100Mb Ethernet segment. (Results were similar for traces across a number of sites.) Two unique packets which are identical up to the specified prefix length are termed a collision. A 28-byte prefix results in a collision rate of approximately 0.00092% in the wide area and 0.139% on the LAN.

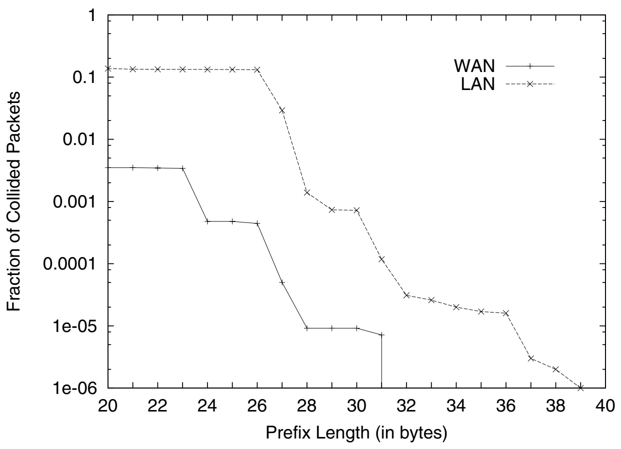

Figure 2: The fraction of packets that collide as a function of prefix length. The WAN trace represents 985,150 packets (with 5,801 duplicates removed) collected on July 20, 2000 at the University of Florida OC-3 gateway [7]. The LAN trace consists of one million packets (317 duplicates removed) observed on an Ethernet segment at the MIT Lab for Computer Science.

Unlike the study reported by Duffield and Grossglauser [4, fig. 4], results reported in figure 2 represent only unique packets; exact duplicates were removed from the packet trace. Close inspection of packets in the wide area with identical prefixes indicates that packets with matching prefix lengths of 22 and 23 bytes are ICMP Time Exceeded error packets with the IP identification field set to zero. Similarly, packets with matching prefixes between 24 and 31 bytes in length are TCP packets with IP identifications also set to zero which are first differentiated by the TCP sequence number or acknowledgment fields.

The markedly higher collision rate in the local area is due to the lack of address and traffic diversity. This result does not significantly impact SPIE's performance, however. LANs are likely to exist at only two points in an attack graph: immediately surrounding the victim and the attacker(s). False positives on the victim's local network can be easily eliminated from the attack graph—they likely share the same gateway router. False positives at the source are unlikely if the attacker is using spoofed source addresses, as this provides the missing diversity in attack traffic, and remain in the immediate vicinity of the true attacker by definition. Hence, for the purposes of SPIE, IP packets are effectively distinguished by the invariant portion of the first 28 bytes of the packet.

3.2 SPIE Architecture

The SPIE system has two parts, a data generation agent (DGA) that is inside or near each router (or a selected set of key routers), and a

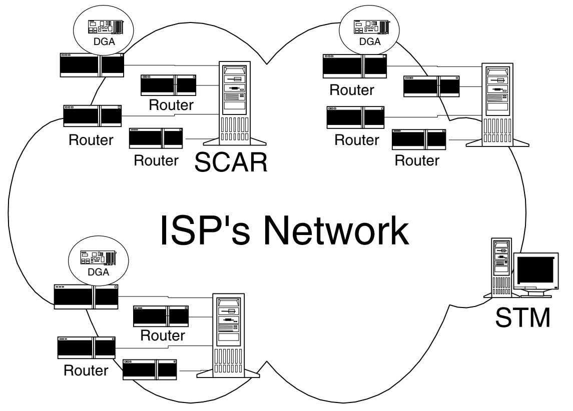

Figure 3: The SPIE network infrastructure, consisting of Data Generation Agents (DGAs), SPIE Collection and Reduction Agents (SCARs), and a special purpose SCAR called the SPIE Traceback Manager (STM).

hierarchical set of SPIE Collection and Reduction (SCAR) agents that ask these DGAs if they have seen the packet in question. The DGA produces the packet digests and stores the digests in the bit-mapped digest tables. The digest tables are stored locally at the DGA for an appropriate period of time, depending on how far back into history a query may need to go. Figure 3 shows the major architectural components of the SPIE system.

Consider a packet traversing a router. The DGA performs a set of independent hashes on the IP and TCP/UDP headers of the packet, producing a set of different hash values. Each of the hash values are used as indices into a bit array, and a “1” is set in each index’s position in the digest table bit array. This is the Bloom Filter mentioned above. Multiple hashes are used to reduce the possibility that two different packets map to the same place in the bit array.

Note that this method will never produce a false negative: If a packet was seen by the router, each of the bits will be set in the digest table, and thus the query will find the bits set. The opposite is not true, however—a DGA may answer “yes” even if it did not see the packet if the bits are set as a result of hashing other packets. This is called a false positive, and the design of the SPIE DGA is carefully tuned to reduce the occurrence of a false positive to a manageable probability.

The rest of the SPIE system is concerned with the query mechanism, attack graph generation, and system security. Since the system is hierarchical, with the network logically broken into regions, the traceback mechanism is scalable to networks of any size. Queries are required to be authenticated, and each SPIE component checks that each entity making a query is authorized to perform the particular query. Queries and their replies are also required to be encrypted. More subtly, since the packet logs are kept as digest tables, there is no way to learn the contents of any specific packet from the digest data.

3.3 Packet Transforms

IP packets are modified during the forwarding process, the two most common modifications being the decrementing of the TTL field and the recalculating of the checksum. However, packets may encounter many more dramatic modifications as they traverse the network—modifications that transform the packet from one form to another.

IP packet *transformation* is the modification of a packet as the result of network layer protocol processing, router error, or malicious intent. SPIE handles only packet transformations whose change of packet state allows for or enhances network data delivery. Such transformations satisfy hardware needs, network management and protocol requirements, and source route requests.

A packet may undergo any number of transformations during network traversal. The transformation of a previously transformed packet is referred to as transform *composition*. Transform composition not only occurs across multiple routers along a forwarding path, but within a single router as well. For instance, routers may encapsulate a packet and then find need to perform fragmentation due to the increased packet length. A packet received by an end host, therefore, may be the result of a combination of transformations.

Attackers engineer IP packet transformations for two reasons, to overwhelm available resources for denying service or taking advantage of protocol implementation vulnerabilities, or to hide within a convolution of changing packet shapes. Distributed reflector attacks [8] are good examples of the first type. The attacker spoofs the source address in an ICMP request packet, or some packet that will cause an ICMP error packet to be generated. The resulting ICMP packet is returned to the spoofed source address—the target of the attack—instead of the real sender of the original packet. When this attack is magnified by thousands of (unknowingly) conspiring computers sending packets that trigger ICMP responses at the same time, the resulting explosion of transformed packets overwhelms the target and denies service.

The second reason for exploiting transforms—to hide—relies on the notion that a packet that changes its shape often, perhaps going through tunnels and being source routed, can confuse a traceback system that expects attack packets to remain largely intact. IP traceback, ingress filtering [5], and other address verification techniques help to mitigate attacks that rely on spoofed addresses, so attackers are having to explore more complicated ways of hiding the source of the attacks.

SPIE supports basic transformations required of RFC 1812-compliant routers, as well as other notable forms of packet transformation.

Source Route

SPIE digesting omits all IP options. Therefore, SPIE is not required to explicitly support transformation of the option fields. The source route options, however, result in the modification of the destination address field within the IP header in addition to the option field, which does require SPIE support.

SPIE’s digesting process is based on the final destination; the digest input for a source routed packet always includes the final destination within the destination field of the IP header, regardless of its true position within the packet. The same method for digesting is used during the traceback of source routed packets.

Fragmentation

To avoid the need to store packet payload, SPIE supports inversion of the first packet fragment only. An attacker cannot control which fragments are received by a victim, assuming that fragmentation occurs within the network itself, so the victim will eventually receive a first fragment to use in traceback. Non-first fragments may be traced to the point of fragmentation which, for fragment-based attacks [6], is the attacker.

ICMP

Fortunately for SPIE, ICMP error packets include the IP header and at least the first eight payload bytes of the originating packet [9]. Thus, SPIE need not record any ancillary data. Ancillary data must be stored, though, to identify the router as the one that generated the error. To invert an ICMP error packet, the original packet is simply extracted from the payload.

IP-in-IP Tunneling

SPIE must handle separately the encapsulation and decapsulation transformations of IP-in-IP tunneling. At the encapsulating router, a record is stored to identify the router as the source of the tunnel, but no ancillary data requires storage. An encapsulated packet is inverted by extracting from the payload the original IP packet.

Inversion at the decapsulating router consists of prepending the encapsulating IP header to the packet presented for traceback. Therefore, sufficient data must be recorded at the point of decapsulation to reproduce the encapsulating IP header.

Network Address Translation

All address and port data required for inversion of packets that undergo NAT processing is already maintained by the NAT translation table residing at the router. To ease memory usage, SPIE takes advantage of the NAT table, recording as ancillary data a reference to the appropriate translation entry of the NAT table. The address and port mappings within the translation entry are used during inversion. Although fairly static, precautions must be taken in the case of a modified NAT mapping.

IPsec

A packet that successfully passes IPsec decapsulation is, with high probability, the exact packet sent by the trusted network at the other end of the tunnel. IPsec explicitly prevents insertion of fake IPsec packets: Packets that are modified or replayed within an IPsec tunnel will not pass IPsec decapsulation. Consequently, IPsec tunnels are essentially additional links between routers in the network topology graph. This fact means that IPsec transforms can be handled by simply noting that the packet was transformed into or out of an IPsec encapsulation, but no additional data for inversion need be kept. The traceback process then treats the other end of the IPsec tunnel as if it were the router's neighbor, and continues the trace from there, in addition to what the physical topology reports as the router's neighbors.

4 SPIE Tap Box

The current router-based SPIE is predicated on the deployment of SPIE-enhanced routers in place of existing routers in the network infrastructure. We understood from the beginning that this deployment path was impractical, but chose a router-based platform as the prototype because it simplified the engineering problems and allowed us to focus on the research of inventing this novel method for single packet tracing. In order to be commercially viable, a SPIE system must be incrementally deployable in the existing network infrastructure without retrofit or "forklift" upgrade. This reality has led to the concept of a *SPIE Tap Box*.

A SPIE Tap Box is a small, special purpose device that implements the full functionality of the SPIE DGA component but without the benefit of access to the router's forwarding engine and internal data structures. Rather, the Tap Box must rely only on the information it can glean by passively tapping the lines into and out of the router.

This approach raises interesting challenges. When SPIE is deployed as a Tap Box rather than inside a router, there are choices about where the Tap Box can be placed with respect to the router. Once possibility is to place it next to the router, so each Tap Box records information for its nearby router. Another approach is to assign a Tap Box to each link, one between each pair of routers on point-to-point links, and one for each broadcast network. This is called the "link as vertex" approach because each Tap Box (vertex) would sit on a network link. Neither approach changes the fundamental SPIE algorithms, however, so the choice is a matter of engineering considerations. Analysis of the models, however, quickly revealed that the link-as-vertex approach exploded the memory requirements, so we have designed and implemented the Tap Box as an adjunct device.

5 Conclusion & Future Work

Developing a traceback system that can trace a single packet has long been viewed as impractical due to the tremendous storage requirements of saving packet data and the increased eavesdropping risks the packet logs posed. We believe that SPIE's key contribution is to demonstrate that single packet tracing is feasible. SPIE has low storage requirements and does not aid in eavesdropping. Furthermore, SPIE is a complete, practical system. It deals with the complex problem of transformations and can be implemented in high-speed routers (often a problem for proposed tracing schemes).

The most pressing challenges for SPIE are increasing the window of time in which a packet may be successfully traced and reducing the amount of information that must be stored for transformation handling. One possible way to extend the length of time queries can be conducted without linearly increasing the memory requirements is by relaxing the set of packets that can be traced. In particular, SPIE can support traceback of large packet flows for longer periods of time in a fashion similar to probabilistic marking schemes—rather than discard packet digests as they expire, discard them probabilistically as they age. For large packet flows, odds are quite high some constituent packet will remain traceable for longer periods of time.

For a more in-depth description of SPIE in general, including packet digesting and the query subsystem, please refer to [12, 14, 15]. Also please see the SPIE website, http://www.ir.bbn.com/SPIE.

References

[1] Bellovin, S. M., Leech, M., and Taylor, T. ICMP traceback messages. Internet Draft, IETF, Oct. 2001. `draft-ietf-itrace-01.txt` (work in progress).

[2] Bloom, B. H. Space/time trade-offs in hash coding with allowable errors. *Communications of ACM 13*, 7 (July 1970), 422–426.

[3] Burch, H., and Cheswick, B. Tracing anonymous packets to their approximate source. In *Proc. USENIX LISA '00* (Dec. 2000).

[4] Duffield, N. G., and Grossglauser, M. Trajectory sampling for direct traffic observation. In *Proc. ACM SIGCOMM '00* (Aug. 2000), pp. 271–282.

[5] Ferguson, P., and Senie, D. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. RFC 2267, IETF, Jan. 1998.

[6] Microsoft Corporation. Stop 0A in tcpip.sys when receiving out of band (OOB) data. `http://support.microsoft.com/support/kb/articles/Q143/4/78.asp`.

[7] National Laboratory for Applied Network Research (NLANR). Network traffic packet header traces. `http://pma.nlanr.net/Traces/Traces`.

[8] Paxson, V. An analysis of using reflectors for distributed denial-of-service attacks. *ACM Comp. Comm. Review 31*, 3 (2001).

[9] Postel, J. Internet Control Message Protocol. RFC 792, IETF, Sept. 1981.

[10] Postel, J. Internet Protocol. RFC 791, IETF, Sept. 1981.

[11] Sager, G. Security fun with OCxmon and cflowd. Internet 2 Working Group Meeting, Nov. 1998. `http://www.caida.org/projects/NGI/content/security/1198`.

[12] Sanchez, L. A., Milliken, W. C., Snoeren, A. C., Tchakountio, F., Jones, C. E., Kent, S. T., Partridge, C., and Strayer, W. T. Hardware support for a hash-based IP traceback. In *Proc. Second DARPA Information Survivability Conference and Exposition* (June 2001), vol. 2, pp. 146–152.

[13] Savage, S., Wetherall, D., Karlin, A., and Anderson, T. Network support for IP traceback. *ACM/IEEE Trans. on Networking 9*, 3 (June 2001), 226–239.

[14] Snoeren, A. C., Partridge, C., Sanchez, L. A., Jones, C. E., Tchakountio, F., Kent, S. T., and Strayer, W. T. Hash-based IP traceback. In *Proc. ACM SIGCOMM '01* (Aug. 2001), pp. 3–14.

[15] Snoeren, A. C., Partridge, C., Sanchez, L. A., Jones, C. E., Tchakountio, F., Schwartz, B., Kent, S. T., and Strayer, W. T. Single-packet IP traceback. *ACM/IEEE Trans. on Networking*. to appear in December 2002 issue.

[16] Song, D. X., and Perrig, A. Advanced and authenticated marking schemes for IP traceback. In *Proc. IEEE Infocom '01* (Apr. 2001).