

Session Based Logging (SBL) for IP-Traceback on Network Forensics

Omer Demir, Ping Ji and Jinwoo Kim

Graduate Program on Forensics Computing
Department of Mathematics and Computer Science
John Jay College of Criminal Justice
The City University of New York
{odemir, pji, jwkim}@jjay.cuny.edu

Abstract

The widely acknowledged problem of reliably identifying the origin of information in cyberspace has been the subject of much research. Due to the nature of the Internet protocol, the source IP can be easily falsified which results in numerous problems including infamous denial of service attacks. The combination of smart devices with powerful processing capabilities once observed only in mainframe computers decades ago and the presence of the Internet which allows communications between all those devices exacerbate the problem. In this paper, we propose a novel technique called Session Based Logging (SBL) for simple and effective IP-Traceback and logging mechanism. SBL is easy to implement and also has significant advantage of saving storage space over previously proposed schemes. Moreover the SBL approach has clear edge under sensitive privacy regulations since it does not need to capture detailed contents of each individual communication session. Experimental results show its potential and ease of execution from free of any agent software installation on the logging machine. The proposed SBL scheme on this paper currently supports only TCP sessions but we believe this approach could be further extended to UDP connections which have many inherent network security problems.

I. INTRODUCTION

The nature of the Internet protocol has been known to make it difficult for reliably identifying the actual source of information in cyberspace. Information over the Internet travels in small units, often called packets, and the source of the information – the IP address of the source host - is contained inside the packet with other data contents. Since the IP address of the *original* source computer is often one of the most critical evidences on on-line investigation, law enforcement agencies and system administrators are encountered with enormous challenges due to this loop-hole in Network Forensics.

IP-Traceback, accordingly, is a method for reliably determining the origin of a packet on the Internet [1]. The major problem of determining the *real* source address of an incoming packet is that the Internet routers may replace the source IP address with their own addresses in network packets, and there is no provision in TCP/IP to discover the true origin of a packet [2]. The consequence therefore is an easy alteration or a masquerade of source address from ill-minded sender for criminal activities in the cyberspace. One of the worst cases that this loop-hole can lead is the infamous *Denial-of-Service* (DOS) attacks. Under DOS attacks, targeted machines suffer significantly from exhausting valuable resources by flooded spoofing messages from senders who often hide their identities by incorrect or altered IP addresses.

In most situations, this kind of criminal activity is extremely hard to prevent and trace once it is committed. Current techniques to trace IP address of a remote machine is far from the one that has been used in telecommunication infrastructure, which is efficient and light-weighted in terms of the amount of information to be logged. Specifically, Telephone logging systems usually include the time of the call, the number of the caller and receiver, the duration of the call and the place of the call. Good topography of relations in the telephone system can be built based on the logged information. However, in Internet, no IP-Traceback technology has adopted the simple approach that is similar to the one used in telephone

auditing systems. In addition, current IP-Traceback logging techniques do not provide long term logging mechanism, and the duration of logging is usually on the order of hours at most. If there were call logs for computer communications, it would be very helpful to law enforcement agencies for later forensics investigation.

In this paper, we propose a Session Based packet Logging (SBL) technique for IP-Traceback that effectively provides log information for long time period (potentially in the order of months), in the meantime, respects the privacy of communications given it does not log the contents of detailed data contents for each communication session. Our SBL approach addresses and solves the problems residing in IP-Traceback from different perspectives in comparison with the previous approaches. We examine the problem by applying conventional schemes for telephone communication system in Forensics Networks. To this end, we adapt existing schema being used in telecommunication investigation to the Internet protocol to realize simple and efficient packet logging. Although the proposed SBL scheme in this paper currently supports only TCP sessions but we believe this approach could be further extended to UDP connections which have many inherent network security problems.

The rest of this paper is organized as following: in Section II, we briefly discuss several related IP-Traceback approaches that tried to remedy discussed problems; In Section III, we propose the Session-Based packet Logging (SBL) scheme in detail; in Section IV, we present our evaluation results that are derived from experiments of SBL; Finally, in Section V, we conclude this paper and discuss potential directions of future work.

II. RELATED WORK

There have been several IP-Traceback techniques. Snoeren et al. [3] present a hash-based IP-Traceback approach called Single packet IP-Traceback. In this technique, a hash based scheme is offered for single packet IP-Traceback and it is claimed to require a space of 0.5% of link capacity for recording. This technique enables single packet IP-Traceback but it has several limitations. First, the original attacking packet has to be found in order to search the audit. The original packet's digest is acquired and then the query is done using that digest. It is not common in Forensic Analysis to have an original copy of an attacking packet. Most of the time we may have only the source IP address and the time of the packet. Another problem is false positives. In case of having the original packet for IP-Traceback, the possibility of false positives will make it difficult to see the log in the court. In order to use them they have to be certain and there shall be no chance of false positives. Memory limitation is another drawback. The amount of time during which queries can be supported is directly dependent on the amount of memory dedicated to SPIE.

Savage et al [4] describes a general purpose IP-Traceback mechanism based on probabilistic packet marking in the network. Their approach allows a victim to identify the network path(s) traversed by attack traffic without requiring interactive operational support from Internet Service Providers (ISPs). Moreover, it is possible to traceback the packet after the incidence finished. They proposed several different marking algorithms but there are some problems with their approach. First problem is a backward compatibility issue since the IP header encoding has several practical limitations and it might negatively impact users that require fragmented IP datagram. Second problem is its effectiveness under distributed attack. The implementation of this technique inherently has serious limitations due to the difficulty in correctly grouping fragments together. Consequently, the probability of misattributing an edge, as well as the amount of state needed to evaluate this decision, increases very quickly with the fan out of an attack. Third problem is with path validation. Some number of the packets sent by the attacker are unmarked by intervening routers. The victim cannot differentiate between these packets and genuine marked packets. Therefore an attacker could insert "fake" edges by carefully manipulating the identification fields in the packets it sends. Fourth problem is attack's origin detection. While this IP-level traceback algorithm could be an important part of the solution for stopping denial-of-service attacks, it is by no means a complete solution. This algorithm attempts to determine the approximate origin of attack traffic. There are a number of reasons why this may differ from the true source of the attack. Attackers can hide their true identities by "laundering" attacks through third parties, either indirectly (e.g., "smurf

attacks” [5] or “DNS reflectors” [6]) or directly via compromised "stepping stone" machines or IP-in-IP tunnels.

Ingress filtering is another approach towards eliminating the ability to forge source addresses. As the name implies the incoming packets are filtered at the routers. Routers filter the packets depending on the source address and the real network segment that the router is connected to. One of the possible problems from this method is that it requires the routers with the knowledge of legitimate and illegitimate traffic. Ingress filtering must be deployed as much router as possible for best and effective filtering. Another problem is that even if ingress filtering were universally deployed at the customer-to-ISP level, attackers could still forge addresses from the hundreds or thousands of hosts within a valid customer network [7].

Generally, IP-Traceback techniques start from the router closest to the victim and independently test its upward links until they determine all of them which are used to transmit the attackers’ traffic. This procedure is repeated recursively on the upward routers until the source is founded. This scheme is generally called as link testing. There are two varieties of link testing schemes, input debugging and controlled flooding. In input debugging, victim is supposed to recognize that it is being attacked and develop an attack signature that describes common features all the attack packets. The victim sends this signature to a network operator. Network operator then installs a corresponding input debugging filter on the victim's upstream output port. This way the router from which this packet has arrived can be determined. The process is then repeated recursively on the upstream router, until the originating site is reached or the trace leaves the ISP's. In case of leaving ISP border, the upstream ISP must be contacted and same procedure must be repeated. Unfortunately, management is a considerable problem for this method. Controlled flooding is a link-testing IP-Traceback technique developed by Burch and Cheswick which does not require any support from network operators [8]. This technique is called controlled flooding since it tests links by flooding them with large bursts of traffic and observing how this affects traffic from the attacker. The inherent problem with this technique is that controlled flooding itself happens to be a denial-of-service attack

Another technique for IP-Traceback is to log packets at routers and then use data mining techniques to determine the path that the packets traversed ([9],[10]). This technique could be quite effective as it allows tracing an attack even long after the attack has completed. But the problem is the storage of these log files. Another potential question which might be raised from privacy advocate group is that the log file might contain sensitive data of peoples’ private conversations.

III. SESSION BASED IP-TRACEBACK

In this section, we propose a light-weighted IP-Traceback scheme - Session Based packet Logging (SBL) - that can be easily deployed at network entities for monitoring TCP traffic. Since storage is one of the major concerns in logging systems of network forensics, SBL is designed to only record the *necessary* contents of a communication session. Specifically, in network forensics, we usually do not need the data transmitted nor the session numbers or the fragments of a transaction. All we need is the source and destination addresses and the time stamps of the communication duration. When we have the traces of illegal actions in the victim system (e.g. IP address), we need only to check the IP addresses against the log to investigate the attack. Since our purpose here is to log the IP addresses and the duration of *communication sessions* between hosts, it is enough to log the TCP sessions by recording the first session establishment packet (SYN packet) and the session termination packet (FIN packet) for each connection. In our SBL scheme, we will log the source IP, Destination IP, incoming port of the router, outgoing port of the router and the time stamps for each logged packet. When keeping the logs, we need to consider the packet encapsulation happened at each network entity (e.g., a router). When encapsulating a packet, the network layer protocol of a router envelope the IP packet sent by another network entity as data payload into a new packet. In this situation the original source IP (e.g., the IP address from the previous hop) is located in the first four bytes of the payload in the encapsulated packet. Therefore in SBL logging approach, we record not only the header information but also the first four bytes of the data payload of each logged packet.

Let us now explore the storage space (in the unit of bytes) that each captured packet may need. For each SYN packet we need to record its source and destination IP addresses which consumes 8 bytes space in total. Meanwhile, as we have discussed, the first four bytes of the data payload should be recorded. In addition, we assume that there are 256 incoming ports and 256 outgoing ports for each network router. Thus we need two more bytes to record the input and output port numbers. Summing over all the bytes mentioned above, we have $8+4+2=14$ bytes of information to log. Moreover, we are interested in keeping track of the time stamp of each packet. Since one day has 86400 seconds which can be represented by 3 bytes with a $1/128$ sec precision, combined with the 14 bytes information, for each packet, we totally need 17 bytes to record the information of interest. Provided that a SYN packet size is 62 bytes and a FIN packet is 54 bytes, the ratio of the logged data size to the total size of a logged packet is “ $17 / 62$ ” or “ $17/54$ ”, which saves the storage space significantly. In our experiments, we will keep the log files with each data entry as shown in Table 1. We use Ethereal [11] to monitor the traffic, where a SYN packet can be observed, when the corresponding “Syn” field is set to be “1”; and a FIN packet can be observed if the “Fin” field is set to be “1”. Figure 1 shows partial contents of packet information obtained by Ethereal, where Figure 1(a) corresponds to a SYN packet, and Figure 1(b) corresponds to a FIN packet. .

Table 1: Sample Table for Logs

Time	Incoming Port	Source IP	First 4 bytes of IP payload	Destination IP	Destination Port
00000.001	2	X.Y.Z.D	X.X.X.X	T.B.G.V	6
00000.012	3	A.B.C.D	Y.Y.Y.Y	E.F.G.H	7

```

.... 0... = Push: Not set
.... 0... = Reset: Not set
.... 1... = Syn: Set
.... 0... = Fin: Not set
Window size: 16384
Checksum: 0xdad2 (correct)
Options: (8 bytes)
Maximum segment size: 1460
bytes
NOP

```

(a)

```

Header length: 20 bytes
Flags: 0x0011 (FIN, ACK)
0... .... = Congestion Window Reduced (CWR): Not
set
0... .... = ECN-Echo: Not set
0... .... = Urgent: Not set
1... .... = Acknowledgment: Set
.... 0... = Push: Not set
.... 0... = Reset: Not set
.... 0... = Syn: Not set
.... 1... = Fin: Set
Window size: 17328
Checksum: 0xd399 (correct)
SEQ/ACK analysis

```

(b)

Figure 1: Sample portion of packet logging file from Ethereal Labs. (a) Observation of a SYN packet; (b) Observation of a FIN packet

Given the packet format of SYN and FIN packets, we define the filter format for SBL session logs as the following: 1. Set SYN bit to “1” and the Ack bit to “0”. This is to get rid of the SYN ACK packets sent by the server back to the client, as it would not provide additional information other than the server’s acceptance of the connection; 2. FIN bit must be set to “1”. A sample capture filter for session logging for ethereal is shown below:

“tcp[tcpflags] & (tcp-syn) = 1 and tcp[tcpflags] & (tcp-ack) = 0 OR tcp[tcpflags] & (tcp-ack) = 0”

In the next section, we conduct experiments to evaluate the SBL logging approach and present the corresponding evaluation results.

IV. EXPERIMENTS AND EVALUATION RESULTS

Table 2: Experiment Data

Experiment ID	# of packets	# of logged packets	size of all packets (KB)	size of logged packets (KB)	duration of experiment (hh:mm:ss)
1	263370	1242	102402	95	24:10:21
2	113036	183	102402	14	1:12:07
3	111939	84	102007	7	2:46:58
4	135588	678	102401	52	15:14:14
5	12016	509	5988	39	1:03:57
6	119	4	40	1	0:00:51
7	146143	1221	102401	94	16:28:27
8	88193	1671	33850	128	3:59:37
9	173895	1044	99423	80	6:52:46
10	2190	35	1025	3	0:39:57
11	1529	15	1025	2	0:10:58
12	2373	40	1025	4	22:47
13	5967	11	1025	1	18:23
14	11204	1	1025	1	6:37
15	8554	8	1025	1	5:03
16	9060	11	1025	1	5:20
17	7178	8	1025	1	3:51
18	2863	10	1025	1	1:02
19	5607	8	1025	1	3:12
20	3898	5	1025	1	1:54
21	7232	9	1025	1	3:42
22	9934	6	1025	1	5:34
23	10490	4	1025	1	6:11
24	11091	19	1025	2	13:05
25	1936	30	1025	3	4:17
26	2031	50	1025	4	8:18
27	2318	70	1025	6	3:57
28	2808	58	1025	5	0:26:34
29	1101	15	473	2	0:29:37
30	151003	439	102401	34	3:42:28
31	29551	115	23946	9	2:17:33
32	105662	2018	54827	154	23:59:54
33	50264	428	11971	33	6:11:21

In our experiments, we utilize Ethereal to sniff network communications between one static residence host and other network end hosts. For each round of experiment, we capture the SYN and FIN packets using the SBL scheme as mentioned in the previous section. We run the experiment for 33 times with varied duration for each run. Ideally, the SBL session logging scheme should be implemented at not only the end hosts but also intermediate routers. However, tabbing fast machines to the network to sniff the aforementioned information is more efficient than putting additional computation burdens on the

routers. Thus, we consider that our experiments using Ethernet based on end-to-end systems are sufficient to investigate the performance of SBL logging. Table 2 shows the experiment results of the 33 experiments that we conducted. For each individual experiment, we record the number of packets captured, the number of logged packets, the size of all captured packets, the size of logged packets, and the duration of the experiment.

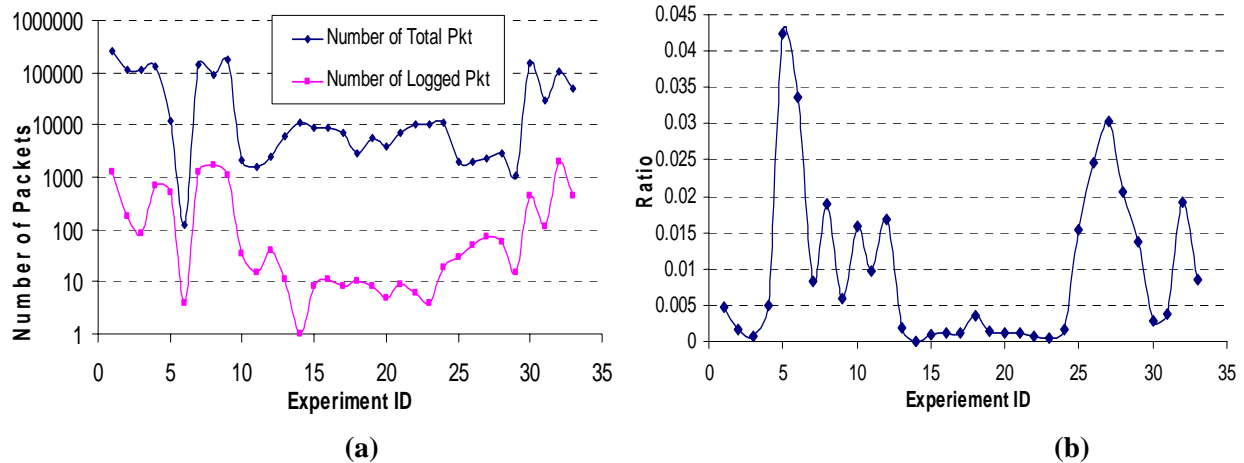


Figure 2: (a) Number of Packet Observed Over Different Experiments (b) The Ratio Between Number of Logged Packets and Total Pkts Over Different Experiments (average ratio = 0.0097)

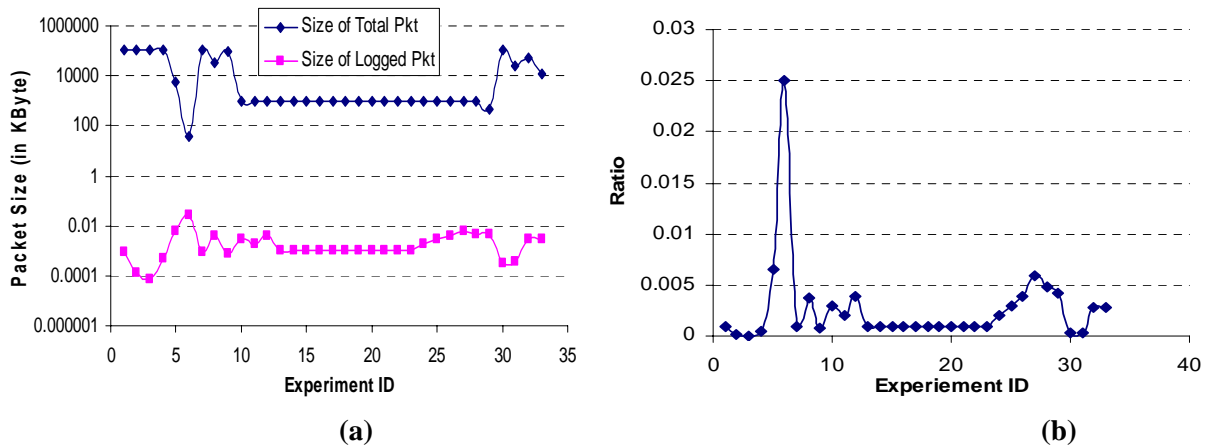


Figure 3: (a) The Size of Packets (in KB) Observed Over Different Experiments (b) The Ratio Between The Size of Logged Packets and The Size of Total Packets Over Different Experiments (average ratio = 0.0027)

To view the experiment results better, we graph them into Figures 2 and 3. Specifically, Figure 2(a) shows the number of total packets vs. the number of logged packet for each run of experiment. Here to display the data clearly, we use log-scale in the y-axis for Figure 2(a). Figure 2(b) shows the ratio of the number of logged packets and the number of total packets in different experiments. Furthermore, Figure 3(a) shows the packet size of total packets vs. the packet size of logged packet in log-scale for each run of experiment, and Figure 3(b) shows the ratio of the packet size of logged packets over it of total packets for each experiment. From the Figures, we observe that the average logged packet number and the average logged packet size are very low (0.0097 and 0.0027 respectively). Even for large amount of total number of packet (or total packet size), the size of logged file remains limited. The maximum ratio

between logged number of packets and the total number of packet is only around 0.043, and the maximum ratio between logged packet size and the total packet size is only 0.025.

As we have observed, the experiment method we develop is suitable for a regular user. However, for large ISPs, the log files may be huge and stored at multiple network entities (e.g., intermediate routers). Nonetheless, from a simple analysis, we realize that using SBL, for large amount of data logging, the storage space consumed by the log files is also reasonable. For example, for the network communication with a total data of 782KB, an experiment with about 110 hours in duration may be needed. Accordingly, an average log size for the experiment should be about $782/110 \times 24 = 171KB$ per day. Since the fraction between the size of logged data and the size of a SYN packet is "17 / 62", the average size of the corresponding log file can be estimated as $171KB \times 17/62 \approx 50KB$. For a 1000 user environment the log file size would be approximately $50KB \times 1000 = 50MB$ per day, and about 1.5GB for a 30 day month. For a 1 Million user environment it would take approximately 50 GB per day and 1500 GB per month. These values are very reasonable the log files of recording session-based traffic for large ISPs. When compression techniques are used, another 2/3 of the storage space can be saved in addition.

V. CONCLUSION

In this paper, we propose a Session Based packet Logging (SBL) technique to log traffic for TCP sessions for network forensics investigation purposes. Compared to related research on IP-Traceback, SBL adopts the data auditing principle used in traditional Telephone networks, which makes SBL easy to implement and have significant advantage in terms of saving storage space. Specifically, the SBL approach only records the IP addresses and the duration of communication sessions by recording the SYN and FIN packets over the logging period. Thus, SBL logs limited yet sufficient information for Network Forensics investigation to deal with network-based criminal cases. Furthermore, the SBL approach does not need to capture detailed contents of each individual communication session, therefore, protects people's privacy very well. Using SBL approach, there is no need to install any agent software on the logging machine. The regular logging mechanism with filtering capacity will work fine.

The SBL approach that we propose in this paper focuses on logging packets in TCP sessions. However, many network security problems reside in UDP connections, and having SBL logs for UDP traffic is crucial for investigating UDP based attacks. In our future work, we would like to extend our research on Session Based Logging scheme in the direction of recording UDP traffic efficiently. Another interesting future research along the line of session based network forensics system is to not only log the useful packets but also mark essential packets along the communication path, so that the session based logs may be stored *only* at end-hosts to relax the burden inside the Internet. Currently, as a future direction, we are also investigating a session-based packet marking scheme, which is an increment of the SBL approach presented in this paper.

VI. REFERENCES

- [1] Wikipedia, www.wikipedia.org
- [2] R. T. Morris, "A weakness in the 4.2BSD Unix TCP/IP Software," AT&T Bell Labs, Tech. Rep.Computer. Sci.117, 1985.
- [3] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Beverly Schwartz, Stephen T. Kent, and W. Timothy Strayer, "Single-Packet IP Traceback", Dec 2002.
- [4] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Network support for IP traceback," IEEE/ACM Trans. Networking", vol. 9, pp. 226–237, June 2001.
- [5] CERT Advisory CA-98.01 "smurf" IP Denial-of-service attacks", Jan 1998.
- [6] CERT Incident Note IN-20004 "Denial-of-service attacks using name servers", Apr 2000.
- [7] CERT Advisory CA-2000--01 "Denial-of-service developments", Jan 2000.
- [8] H. Burch and B. Cheswick, "Tracing anonymous packets to their approximate source" in Proc. 2000 USENIX LISA Conf., pp. 319-327, Dec. 2000.
- [9] G. Sager, "Security Fun with OCxmon and cflowd," presented at the Internet 2 Working Group, Nov 1998.
- [10] R. Stone, "CenterTrack: An IP overlay network for tracking DoS floods," in Proc. 2000 USENIX Security Syrup., pp. 199-212, July 2000
- [11] Ethereal Packet Sniffer, <http://www.ethreal.com>