A Method to Compress and Anonymize Packet Traces

Markus Peuhkuri

Abstract— **Data volume and privacy issues are one of problems related to large-scale packet capture. Utilizing flow nature of Internet traffic can reduce data volume. Removing sensitive information such as IP addresses enchanges privacy. Our method makes possible to have same replacement value for given IP address even if capture location or time is different.**

Keywords—**Packet capture, data compression, anonymization**

I. Introduction

PACKET capture from live networks is one of basic tools for network research. The packet traces can be used to study how real users, applications, and implementations behave. Potential study population in packet trace studies can be very large – the whole Internet. Using other approaches, such as instrumenting end systems, large study populations are difficult to organise.

There are two main problems in packet trace collection and analysis: volume of data and privacy issues. Even if the capture device can keep with data flow, subsequent storage may be a problem. An efficient and fast method to compress data is useful.

The network traffic carries potentially sensitive information such as passwords or other identification information. While increasing use of encrypted connections (IPSec, TLS, and SecSH) protects the payload, even the knowledge that there exists a communication between two parties can be sensitive.

The legislation varies from one country to another and it is not always clear if something is permitted or not as legislation cannot not refer to a particular technology but speaks on generic terms.

M. Peuhkuri works at Helsinki University of Technology, Networking Laboratory, Espoo, Finland. E-mail: Markus.Peuhkuri@hut.fi. Phone: +358-9-451 2467. Fax: +358-9-451 2474.

This work is supported by Academy of Finland under contract 1163846.

To conduct large-scale measurements, a researcher needs co-operation with network operators as they control the core network: a very interesting measurement location. In many cases the operators are not interested other measurements that for ones that directly support their network operations. Legal issues, customer trust, operations secrets, and network reliability requirements are reasons for an operator to deny researcher access to the network. For the first and the second reason a proper desensitization is needed. Designing a packet capture equipment and measurement setup properly can solve the last one.

In this paper we first study efficient compression of packet trace data. In Section III we look at Internet headers and evaluate how sensitive each field is. In Section IV we present our method remove sensitive information from packet traces. We shortly introduce our equipment in Section V. Finally we conclude properties of our methods combined.

II. Flow based compression

Network traffic capture with current link speeds needs much of storage. For a network monitoring aggregate metrics can be calculated and data reduced even further as the needed information is known in advance. For research purposes, however, it is not always known in advance what information is important. If possible, all of the header data is saved. However, there are many fields in the IP, UDP, and TCP headers that do not change over the lifetime of a connection when observed at a single location.

The method described in [1], [2] is to code only the difference between consecutive packets in a same flow and use a short code for default (in-order delivery) case. The method is intended to be used where number of simultaneous flows is small, e.g. on first-hop links. However, we can use a larger identifier space to identify more simultaneous connections, as we are not limited by header space. We take the source and destination IP address[1], the protocol number and, in case of UDP and TCP, also the port number, as these define a flow at datagram level [3]. We calculate a hash value based on these identifiers and use it as index to a table.

Table size depends on available RAM at capture device.

[1] Possible address anonymization (Section IV-A) happens before this.

Our current file format allows maximum size of 2^{24} that requires 1 GiB of RAM with 64 byte packet length.

The packet is compared with the one in that table position. If the packet belongs to a different flow then the one in table entry (hash collision, or previously vacant position), the new packet is inserted into the table and a new flow record (packet headers desensitized, see Section IV) is written to the output stream. If the packet belongs to the same flow, we compare the packet in table to this one.

At first the version, header length, type of service (or DS-byte) and time to live (TTL) fields at IP headers are compared. If there is a difference, then the packet is written on stream as in non-matching flow. If, however, the IP datagrams are equal, then upper protocols are studied.

TCP If a segment contains in-sequence data, does not acknowledge new data and is same size as preceeding, only a case code and a flow id (total 32 bits) is recorded. Other special cases include if either sequence or acknowledgement numer is within 32 KiB range. If TCP flags are different, then the entre TCP header is stored excluding checksum. TCP options are saved as such as there is no efficient way to compress selective acknowledgment (SACK) [4] or time stamp [5] options. The end of options (EOP) and no operation (NOP) options [6] are removed.

UDP Equal length datagrams are recorded with a case code and a flow id. If there is a difference in length, also length is recorded.

ICMP An ICMP flow exits if ICMP type and code equals. In addition, for ICMP types including IP datagram the compression is applied also on the datagram in payload. For parameter problem and redirect messages also a pointer or a gateway address are stored.

Similar methods are used to store IGMP and IP-in-IP flows. Information related to IP fragmentation (identification, flags and fragment offset) is not stored by default. However, it is possible store all identification information (adds 32 bits for every datagram) or information for only those datagrams which allow fragmentation (adds 40 bits for fragmented datagrams and 24 bits non-fragmented).

Time information is saved with a microsecond resolution. This is stored with variable-length coding: if the time difference to the previous packet (may belong to another flow) is less than $2^{15}\mu$s (≈32 ms) or less than $2^{31}\mu$s (≈2 s) it is coded with 16 or 32 bits. If the time interval is longer, it is coded with 96 bits. If a greater granularity is needed, it is trivial to change the base unit to nanoseconds, in which case the limits change proportionally.

In an optimal case, where an in-sequence TCP segment arrives within 32 ms of preceding, no hash collision happens and no fragmentation information is recorded, the packet is stored into 48 bits, 6 bytes in contrast to 48 bytes without compression (40 byte header with 8 byte timestamp).

III. What is sensitive in Internet protocols

If we look at IP header [7], most of the fields are non-sensitive. There are only two fields that actually carry sensitive data: the sender's address and the recipient's address. They identify communicating parties to host granularity. In many cases, this is the same as to a single person (or one's family) and thus it is a *person identification* that should not be revealed.

The checksum field can be sensitive, as if all other than address fields are known, it is possible to rule out some set of possible IP addresses. Based on a time-to-live (TTL) value, it is possible to guess how many hops the IP packet has traveled as implementations use values of 32, 64, 128 and 255 for TTL. The total length of IP datagram can give some information about upper protocols or payload. However, the information in these fields cannot be considered sensitive, as there are too many possible matches.

The UDP [8] port numbers are used to identify application. Based on the information we can answer to question: "is somebody using certain application in this network". Again, it is dependent on the number of users in network whether this information is sensitive.

The same discussion about port numbers applies also on TCP [6] as they have same functionality. Other fields in TCP header are related to connection setup or flow control and do not reveal any other information than bytes transferred over connection. A urgent data pointer may carry some application semantics.

The payload carried in UDP or TCP packets must be considered sensitive as the application data may contain for example a private email. This information must not be revealed.

To conclude, both the IP addresses and the payload are sensitive information. The checksum field in short UDP or TCP packets may also have some sensitive information in it if one guesses other fields.

IV. How to anonymize network addresses

There are several approaches to solve the privacy problem. One commonly used method is to replace the IP addresses in network traces with random addresses which are recorded on a table. Once the remapping is done, one cannot know which fake address corresponds to which real address. This approach has, however, several drawbacks:

- topology information is lost unless a separate index for each network is maintained,
- there is no mapping between subsequent traces unless the table of mappings is stored *securely*,

- it is not possible to correlate traces collected from different points of network, and
- the table may grow large and thus difficult to store, especially in a capture device.

To overcome these problems, we designed a new cryptography based solution. A brute force search over whole IPv4 address set is feasible. The algorithm must be selected carefully for this reason. We selected symmetric encryption because of speed; the system is described below.

A. Our solution to sanitize network addresses

Our system takes at maximum a 1024 byte secure key. If we just want to get a single trace without mapping between different traces, we can use `/dev/random` or a similar source of random bits. A 128-bit key for Blowfish [9, p. 336] is generated using MD5 [10] over the supplied key (Fig. 1).

Each time an IP address is seen in a datagram, either in the IP header or in some other location, such as in ICMP message, it is scrambled. A table similar to a routing table is consulted to find "hostpart", i.e. how many bits should be scrambled; the "network part" is left unencrypted. The original IP address is concatenated with a 32-bit block of the key and encrypted into a 64-bit value using Blowfish in electronic codebook (ECB) mode and the low part of the encrypted value is used as an index to the hash table. If there is no entry at that location or the encrypted value is different, a record is written into the packet output stream with the top 24 bits of real address[2] and encrypted value, total 96 bits (12 bytes).

Each IP address in a datagram is replaced with value, which has the topmost 8 bits from the original address and lowest $\lceil \log_2 tablesize \rceil$ bits from the encrypted value.

When the compressed and scrambled stream is expanded, the encrypted values are used as a key to the database where the random IP addresses within each network are generated. If there is no database entry for the encrypted value, a new free random IP address within the network is selected and used as replacement for the encrypted value. A single encrypted IP address maps always onto same replacement value as the database is disk based and persistent. This makes it possible to keep one-to-one mapping between different traces and there is no need for any database functionality in the capture device. The measurement device can be seen as a filter that takes IP traffic and produces a privacy-protected stream of packet headers. An example of file contents is shown in Fig. 2.

[2]The network part can be shorter than 24 bits, in that case lower bits are zero.

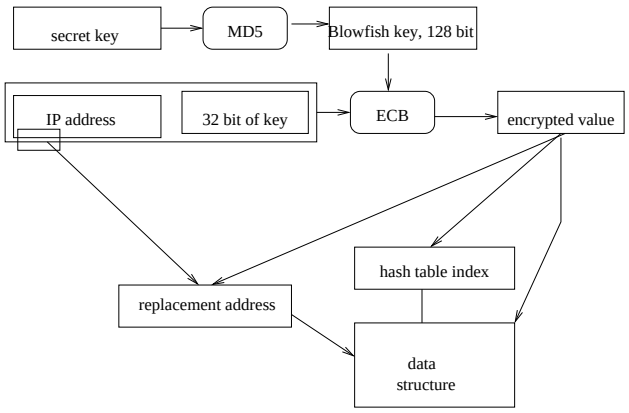


Fig. 1
SCRAMBLING OF IP ADDRESSES

byte	contents	description
0	Enc: 10.2.3.0: 71ee1...	IP address (10.2.3.4) scramble info
12	Enc: 192.168.0.0: b64a...	(192.168.5.2)
24	t=971702544.237632: IP: TCP	TCP segmement from 10.2.3.4 to 192.168.5.2
70	Δt=0.001432: TCPinSEQ: flow=342412	In-sequence TCP
76	Enc: 192.168.0.0: 4af0...	(192.168.3.145)
88	Δt=0.432235: IP: UDP:	IP/UDP datagram
112	Δt=0.000966: TCPinSEQ: flow=342412	In-sequence TCP
118	...	...

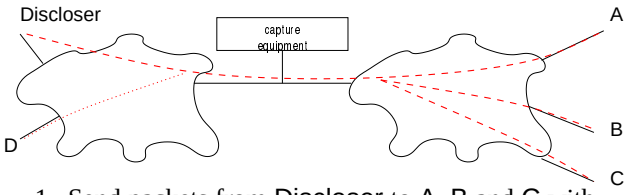
Fig. 2
EXAMPLE OF TRACE FILE CONTENTS WITH ENCRYPTED IP ADDRESS AND FLOW COMPRESSION

The database used for one-to-one re-mapping of IP addresses contains no sensitive information and thus does not need any special handling. The only data that must be protected is the secret key used in capture devices.

B. Possibility to address disclosery

The address scrambling has one weak point, namely it is vulnerable to chosen plain text (in this case chosen IP address) attacks. If the Discloser knows that there is an on-going measurement and one knows that the trace will be published, one can find out mapping between selected IP addresses.

The Discloser sends some, possibly handcrafted, IP packets in some defined pattern; as the port numbers are left intact, one may utilize them to carry extra information. The aim is to be able to locate packets from a published trace. As one locates the packets (maybe using time



1. Send packets from Discloser to A, B and C with some identifiable pattern.
2. Send packets from Discloser forging D as source address to A, B and C with some identifiable pattern.
3. Locate sent packets from trace by taking into account that some packets may be lost.

Fig. 3
CHOSEN IP ADDRESS ATTACK

stamps as helper) the value corresponding one's IP address is found.

When the Discloser has supplied enough packets to learn its own IP address, one can send packets to those hosts one wants to learn about. The Discloser can learn any IP address, if the packets to wanted destination travels over measurement point. It may be possible also use forged sender address to learn IP addresses on same side of measurement point unless strict network ingress filtering is enforced [11]. See Fig. 3 for description.

One must note, however, that this attack is applicable also for the popular sequential replacement approach.

C. Implementation issues

There are some practical issues regarding real security of packet traces. One problem is that the encryption key is kept in memory all the time and may get written on swap disk. We can mark the memory location containing the keys non-swappable if we run capture routine as root user. The measurement system is a real-time one that should be dimensioned so that there is no need to use disk for virtual memory to guarantee acceptable performance.

The selected key size (128 bits) is considered to be safe for now. In future, there may be also need to make key length longer. As an extra measure, a part of the secret is appended to data to be encrypted as "salt". If there is some way to do brute force search, this should make it a bit harder.

V. IMPLEMENTATION

The capture system is implemented on a PC with Celeron processor with 512 MiB of memory and 4 SCSI3 disks running Linux 2.2. Measurement cards are two DAG3 cards by Waikato University (NZ) [12]. The cards are set up to capture two first cells of every PDU. Two-cell capture is needed to record the TCP flags in the LAN Emulation network [13, p. 51] [7], [6].

The hash table size for compression is 524,287; the same table size is used also for address anonymization. This allows enough memory for I/O buffering.

We studies efficiency of compression and desensitization by taking 56 hour network traces measured daytime between Networking laboratory and the rest of campus. Total 46,587,460 packets were captured (TCP: 45,068,043; UDP: 1,432,800; other 86,617). For practical reasons, the test data was read from disk file. File sizes and corresponding times are described in Table I. With compression "plain" equivalent information (expect address scrambling) is saved but no flow compression is utilised. Test was performed on Solaris machine with Ultra SPARC 300 MHz processor. Network sizes were traditional A, B, and C classes expect for the campus network (B-class) 24-bit networks were used.

TABLE I
FILE SIZES AND PROCESSING TIMES

Compression	Size [MiB]	Time [s]	Pkts/s
none	4,886	-	-
gzip	2,218	5,108	9,120
anon+flow	770	1,374	33,906
anon+flow+gzip	318	2,023	23,028
plain	1759	-	-

Based on estimated packets per second figures presented in Table I, the flow compression and desensitization is feasible at speeds of 155 Mbit/s with current low-end processors. The whole system with Dag cards was not tested because of malfunction of a traffic generator.

In future we also study feasibility of including RTP information into compressed data but there exists a privacy issue as a partial payload of UDP datagram is recorded in case we erroneously identify non-RTP packet as RTP packet.

VI. CONCLUSIONS

We have presented a system that helps to solve two problems in large-scale packet capture: volume of data and privacy. Compression utilizes the flow nature in Internet traffic to reduce data volume while preserving as much as possible interesting information for network research. While packet capture is always competition between network and computer speeds, the system helps with the bottleneck: from RAM to persistent memory devices.

For a long-term storage one can also use general-purpose data compression. The non-flow compressed file yields better compression ratio because of greater redun-

dancy, but still the advantage is of the order of 1:2.

IP address anonymization reduces risk of address disclosure even if it is vulnerable for at least one type of attack. This level of protection should make it easier to researchers to exchange measurement traces even if measurent traces are not made available for public.

REFERENCES

[1] V. Jacobson, "Compressing TCP/IP headers for low-speed serial links," Request for Comments RFC 1144, Internet Engineering Task Force, Feb. 1990.

[2] M. Degermark, B. Nordgren, and S. Pink, "IP Header Compression," Request for Comments RFC 2507, Internet Engineering Task Force, Feb. 1999.

[3] K.C. Claffy, H.-W. Braun, and G.C. Polyzos, "A parameterizable methodology for internet traffic flow profiling," *IEEE Journal on Selected Areas in Communications*, pp. 1481 – 1494, Oct. 1995.

[4] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options," Request for Comments RFC 2018, Internet Engineering Task Force, Oct. 1996.

[5] V. Jacobson, R. Braden, and D. Borman, "TCP Extensions for High Performance," Request for Comments RFC 1323, Internet Engineering Task Force, May 1992.

[6] J. Postel, "Transmission Control Protocol," Request for Comments RFC 793, Internet Engineering Task Force, Sept. 1981.

[7] J. Postel, "Internet Protocol," Request for Comments RFC 791, Internet Engineering Task Force, Sept. 1981.

[8] J. Postel, "User Datagram Protocol," Request for Comments RFC 768, Internet Engineering Task Force, Aug. 1980.

[9] Bruce Schneier, *Applied Cryptography Second Edition : protocols, algorithms, and source code in C*, John Wiley & Sons, Inc., 1996.

[10] R. Rivest, "The MD5 Message-Digest Algorithm," Request for Comments RFC 1321, Internet Engineering Task Force, Apr. 1992.

[11] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing," Request for Comments RFC 2827, Internet Engineering Task Force, May 2000.

[12] "The dag project," On web, `http://dag.cs.waikato.ac.nz`, Referred 2001-06-26.

[13] Technical Committee, "LAN emulation over ATM version 2.0 — LUNI specification," Tech. Rep. af-lane-0084.000, The ATM Forum, July 1997.