TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 0?/?? pp???-???
Volume 18, Number 2, April 2013

TIFAflow: Enhancing Traffic Archiving System with Flow Granularity for Forensic Analysis in Network Security

Zhen Chen, Ling-Yun Ruan, Junwei Cao, Yifan Yu, Xin Jiang
Department of Automation
Department of Computer Science
Department of Electronic engineering
Research Institute of Information Technology (RIIT)
Tsinghua National Laboratory for Information Science and Technology (TNList)
Tsinghua University, Beijing, China

Abstract: The archiving of Internet traffic is an essential function for retrospective network event analysis and forensic computer communication. The state-of-the-art approach for network monitoring and analysis involves storage and analysis of network flow statistic. However, this approach loses much valuable information within the Internet traffic. With the advancement of commodity hardware, in particular the volume of storage devices and the speed of interconnect technologies used in network adapter cards and multi-core processors, it is now possible to capture 10 Gbps and beyond real-time network traffic using a commodity computer, such as n2disk. Also with the advancement of Distributed File System(such as Hadoop, ZFS etc.) and Open Cloud Computing platform(such as OpenStack, CloudStack and Eucalyptus etc.), it is practical to store such large volume of traffic data and fully in-depth analyse the inside communication within a acceptable latency. In this paper, based on well-known TimeMachine, we present TIFAflow, the design and implementation of a novel system for archiving and querying network flows. Firstly, we enhance the traffic archiving system named TIFA with flow granularity, i.e. supply the system with flow table and flow module. Secondary, based on real network traces, we conduct performance comparison experiments of TIFAflow with other implementations such as common database solution, TimeMachine and TIFA system. Finally, based on comparison results, we demonstrate that TIFAflow has a higher performance improvement in storing and querying performance than TimeMachine and TIFA, both in time and space metrics.

Key words: Network Security, Traffic Archival, Forensic Analysis, Phishing Attack, Bitmap Database, Hadoop Distributed File System, Cloud Computing, NoSQL.

1 Introduction

With the rapid development of the Internet over the last forty years, the Internet has played an increasing important role in our daily lives. Meanwhile, the openness of the Internet has also led to a large number of attacks. It is therefore very important to secure networks by analyzing the network traffic. The misconfiguration of routers can result in disastrous consequences, and network attacks can cause network breakdown, service crashes, and even communication interruptions. The dissemination of unsolicited information and illegal behaviour also affects Internet users normal activities, and there is an underground economy that is based on Internet Scamming and Fraud. These attackers conduct more and more e-crimes, such as spams and phishing attacks on innocent victims etc.

For example, phishing attacks [1-3] are practical problems due to the sensitive information stolen (e.g. monetary user account name and password) and it is

• Zhen Chen* and Junwei Cao are with Research Institute of Information Technology and Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing 100084, P. R. China.
E-mail:zhenchen, jcao@tsinghua.edu.cn

• Linyun Run was with Department of Automation, Research Institute of Information Technology and Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing 100084, P. R. China. Now he is with Department of Computer Science of Purdue University, West Lafayette, IN, USA.
E-Mail: rlyswf@gmail.com

• Yifan Yu is with Department of Electronic Engineering and Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing 100084, China.
E-mail: yuyf10@gmail.com

• Xin Jiang is with Department of Computer Science & Technologies, Research Institute of Information Technology and Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing 100084, P. R. China.
E-Mail: jiangxin_thu@sina.cn

∗ To whom correspondence should be addressed.

Manuscript received: 2013-1-15; revised: 2013-7-15; accepted: 2013-7-15.

estimated that there is about one billion dollars in accumulated losses annually. The operations of both users and financial institutions such as e-banks and e-pay systems have been impaired by phishing attacks.

Research on the life cycle of phishing web sites shows that phishing URLs are temporary, which makes the collection of forensics difficult [4]. The problem is worsened because most innocent Internet users are unaware of a phishing attack that is taking place.

Traffic measurement and real-time analysis [5-7] are basic methods used to detect and prevent such attacks. However, as these attacks became increasingly sophisticated, new attacks are often undetected in time; therefore, traffic archiving technologies[8-13] for future forensic analysis are key to identifying and deterring the attackers.

The implementation of practical traffic archiving technology has encountered some challenges[14]. Currently, the speed of Internet access links is up to 10100 Gbps. This means that the amount of data generated per second may be about 110 GB per link. To record high speed flows, the cost will be rather high, which is a big challenge for most network operators. Meanwhile, many companies either maintain their own networks or delegate their servers to IDCs (Internet Data Centers). Real data traces have shown that there exist a large number of companies whose Internet access bandwidths vary from 100 Mbps to 10 Gbps. The management of these local networks is also a big problem. Therefore, it is necessary to design and implement a low-cost network-traffic archiving system for such enterprise networks.

The other challenge of such system design is how to speed up the traffic query function as a large volume of traffic data needs to be explored. TimeMachine[12,13,15], TM for short, is a traffic collection and indexing system, which can collaborate with IDS like Bro to scan the collected traffic in the past to find the former attack which has not been detect at that time. It works as if it can travel in the time as the name suggested.

In the previous work TIFA[16] is a system where TM is complemented with FastBits packet indexing [17] to provide more flexible query functions, and TMs packet capture and storage management module remain unchanged, this is why the name TIFA (**TI**memachine+**FA**stbit) origins.

The main contribution of this paper is to present TIFAflow, a flow granularity archiving and querying system, and introduce its design and implementation. We enhance flow granularity in TMs storage/indexing management by rewriting TMs indexing and storage module. Based on real network trace, compared with TM and TIFA, we conduct several groups of experiments and detailed analysis. With the acceptable trade-off in indexing operation, it shown that TIFAflow can reduce the storage volume and speed up the flow querying operation in both time and space metrics.

The rest of this paper is organized as follows: Section II describes related works about traffic archival system design and enabled techniques. Section III presents the system design and implementation of TIFAflow. Performance evaluation is conducted and the results are analysed in section IV. Section V introduce the TIFAflow and the experiments compared with TIFA and TimeMachine. Finally, Section VI concludes the paper.

2 Related Work

There are two categories of studies into the recording and storing of network flows: The first involves the recording and querying of statistical information of network flow, and the other is the recording and querying of raw network traffic.

The former focuses on recording and archiving the statistics of network flow information, e.g., the five-tuple (source IP address, destination IP address, source port, destination port, and transport layer protocol), the size of network flow, generation time, and duration.

The latter focuses on the entire network traffics in a monitored network, and records the content of network traffics for forensic analysis of network events. The most challenging problem faced in this case is the capture and storage of the packets arriving in wire-rate, and the indexing of the traffic for further analysis.

With the advancement of large volume storage devices, high speed network adapter cards, and multi-core processors, it is practical to capture 10 Gbps and beyond real-time network traffic with a commodity computer for future traffic analysis. Table 1 presents a summary of a traffic archiving method and system.

Table 1 A summary of traffic archiving and analysis method and system

Scheme	Raw file	Relational database	Special designed database
Pro	Archiving with wire-rate	SQL query support	High Efficiency
Con	Poor query interactivity	Low efficiency	Depend on implementation
Typical system	TM/Nfdump/ OSU flow-tools	Neye/Combi ning	Gigascope/Hyperion/ Bitmap Database/ Tribeca
Developm ent	Medium	Easy	Hard
Method	Indexing module need to speedup querying	Using on-the-shelf module	Storing and indexing module need to be implemented

2.1 Archiving Traffic Statistics Information

Currently, NetFlow [18] and sFlow [19] are the two widely used industrial network standards used to describe network flows. The goal is to establish a system to record, store and query flow information, which can work for network monitoring application.

Generally, relevant research can be divided into three categories according to the method of storing traffic information:

a) Raw file

The advantage of using the raw file-based scheme is the increased speed with which raw traffic is recorded. The disadvantage is the lag behind the real-time traffic as traffic queries in real-time are not supported by this solution. For example, suppose that a certain event occurs at time t, but it can only be detected event until t + delta, for some non-negligible delta.

Another disadvantage of this solution is the absence of indexes of information. To analyse the traffic information, the packets need to be individually retrieved in the raw file, which can make the query latency unacceptable. nfdump [20] and OSU flow-tools [21] belong to this type of work.

b) Common relational database

The advantages of using a common database include security, stability, and support of flexible SQL query statements without additional development. Common relational databases are widely used for various types of data access. However, for specific network areas, it may not be a good choice because it fails to fully explore the features of the application data. Neye [22] and Combining [23] belong to this type of work.

c) Special designed database

Designing a special database for a specific area such as the archiving of a network flow can optimize the performance by utilizing the time and space characteristics for the data storage and query. Gigascope [8] built a high-performance network flow information database. On a dual-core server with a 2.4 GHz CPU, it can process 1.2 million packets per second. Besides, it provides easy, flexible, and imitation SQL query statement syntax GSQL. Tribeca [9] proposes its own query system and the corresponding query statement format. Luca Deri et al. [10,24,25] propose to use a bitmap index database for information storage and querying. It also achieves a better performance than the common relational database.

Francesco Fusco et al. [26] describe the design of a novel multi-core aware packet capture kernel module that enables monitoring applications to achieve high performance packet capture on modern commodity hardware. They also introduce the design and implementation of NET-FLi[14], a high-speed on-the-fly compression, archiving and retrieval of network traffic information.

2.2 Archiving Raw Network Traffic

Three types of implementation are also used to archive network traffic: hardware solution, system-level solution, and application-level method. Table 2 presents the implementation of these three classifications.

a) Hardware-level method

Based on specific customized hardware, such as Cavium Networks OCTEON 58XX [27,28] and Tilera

Table 2 Three implementations of traffic archiving and querying for raw network traffic

Scheme	Typical system
hardware-level	Intel IXP Network Processor[5,6]/Cavium Networks OCTEON 58XX[27,28]/ Tilera TILE Pro[29]
system-level	Gigascope[8]/Tribeca[9]/Hyperion[11]
application-level	tcpdump/ Nfdump [20]/OSU flow-tools[21]/TM[12,13]

TILE Pro 64 manycore Network processor [29,30], traffic capture can be achieved with better performance with 10 Gbps and more. However, it is still required to archive the captured traffic into persistent storage and indexing for querying. This method has a relatively higher cost with better performance.

b) System-level method

The system-level solution is based on the operating system(OS), and implements network traffic archiving and querying. As an example, Hyperion [11] can record more than 1,000,000 packets on a common computer, while supporting query performance, and ensuring 200,000 packets per second for the index. The stream-oriented file system invoked by Hyperion can ensure continuous disk reading and writing to achieve the highest write speed.

c) Application-level method

The application-level solution involves building a network traffic archiving and querying system based on common operating systems, such as Linux or FreeBSD. This method allows the system to work well with other compatible networking functions support by the OS. TM is an application-level method designed for high speed network traffic recording and queries. TM uses a cutoff scheme to reduce the volume of the network traffic without impairing the amount of the network flow information, and can be deployed and work with Bro-IDS [31].

3 System Design

A typical high speed traffic archiving and querying system has three critical functions: packet capture, packet indexing and storage management. Packet capture in wire speed has previously been explored in Refs.[10, 12, 26], while packet indexing and storage management have also been investigated in Refs.[11-13,17,24,25,32].

3.1 Genealogy

As indicated in the operating experience with TIFA [16], efficiency-related problems occurred in packet granularity based storage management in the traffic archival system. First, maintaining an index for each packet will cause larger space consumption; secondly, most of the flow level queries need to aggregate the individual packets belong to the same flow on-the-fly, causing a much longer response delay. To address these problems, an intuitive solution is to organize the storage management with flow granularity.

Based on this idea, the new design is explored and the performance evaluation is conducted to demonstrate the benefits. The key novelty is that we re-implemented TIFA TM with flow granularity in its indexing, storage and query module, and indexed the flow with FastBit to provide more flexible query functions. In addition, the packet capture module and storage management module of TM were reused in our design. Several groups of experiments are conducted to show that this novelty can reduce the cost of the storage volume and speed up the flow querying operation. Figure 1 presents the result of the existing initiatives in TIFA flow system.

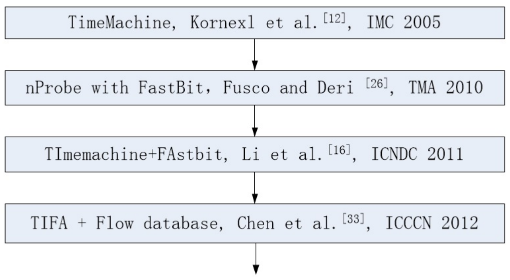

TimeMachine, Kornexl et al.[12], IMC 2005
nProbe with FastBit, Fusco and Deri [26], TMA 2010
TImemachine+FAstbit, Li et al.[16], ICNDC 2011
TIFA + Flow database, Chen et al.[33], ICCCN 2012

Fig. 1 Genealogy of TIFAflow system

3.2 TM overview

TM truncates the data flow based on the heavy-tailed characteristics of Internet traffic, which can significantly reduce the amount of data to be stored, while retaining enough network information.

Eight threads were implemented in TM to run the independent function block. Figure 2 presents the structure and work principle of TM. One thread is run for packet capture and classification, one thread is run for listening to the UI input and output, and one thread is run for statistics and logging functions. Four threads are run for four separate types of indexing function according to the structure shown in Figure 3, which are

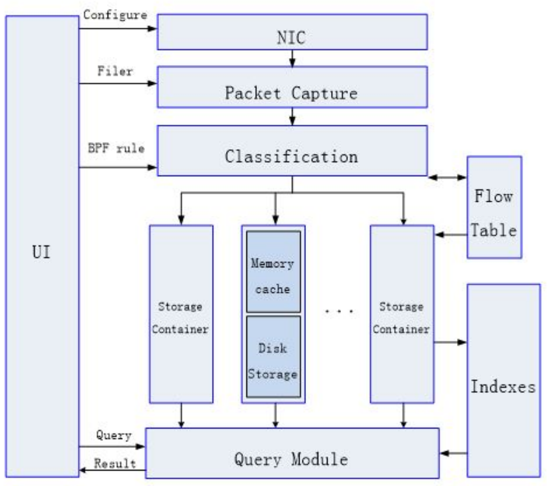


Fig. 2 Structure and work principle of TM.

Singleton, Pair, Four and Five Tuple indexing thread. It need to point out that TM is packet based indexing and storage. The last thread then takes charge for index aggregation.

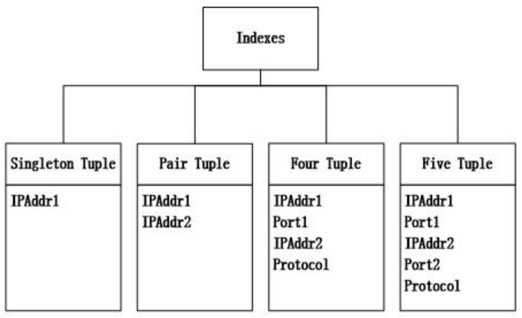


Fig. 3 Four type of indexes in TM.

TM can cache large amounts of data flow spanning several days. It stores the flow data after they are truncated. It also provides an efficient query interface to retrieve real-time data packets and automatically manages the available storage space. It also depends on the strict surveillance of characteristics associated with heavy-tailed network traffic, and can record the highest number of completed connections.

3.3 TIFA overview

TM enables the realization of traffic archiving, flow truncation, and query operations for trace files. There remains areas for improvements to make traffic archival more useful and efficient.

TIFA [16] is designed to integrate TM with Fastbit to improve query performance.TIFA provides more flexible query functions to enhance TM with FastBits packet indexing where TMs packet capture and storage management module are used.

Table 3 Fileds Indexed for each packet with FastBit

Sip	Dip	Sport	Dport	Protocol	Offset	Time

FastBit [17] uses a bitmap index with an SQL interface to speed up archiving and querying for large amount of data. the vertical structure and the compressed bitmap directory are the key for fast archiving in FastBit.

FastBit can build indices for data provided by columns, and each column is a file. Usually, related files are placed in the same directory. Figure 4 shows the structure and work principle of TIFA.

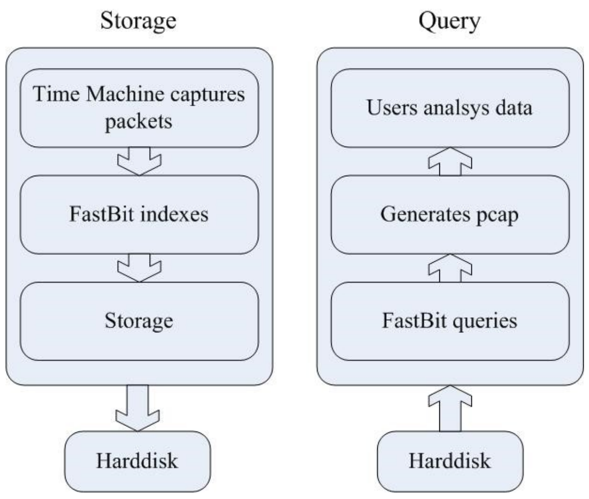


Fig. 4 TIFAs structure and work principle

The indexes' fields for each packet are indicated in Table 3. Field sip, dip, sport and dport represent source IP address, destination IP address, source port, and destination port respectively. Protocol is used to identify it as either TCP or UDP data. Each dump file is limited to a size of 500 MB. Offset indicates the offset of each packet in the file. Figure 5 shows the principle of FastBits packet indexing used in TIFA, the indexes are finally stored in tables of FastbBit.

Collection module

The collection module intercepts traffic through the system IO. Whether the packet is captured by the network interface or is read from the traffic trace file is determined by the user's configuration. When a packet is captured by the collection module, a signal is transferred to the flow table module updating the flow information in the flow table. Then the collection

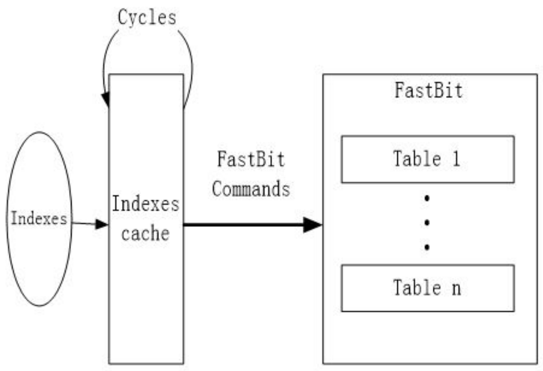


Fig. 5 Indexes with FastBit in TIFA.

module decides whether to transfer the packet to the flow index module based on the processing results of the flow table module.

Flow table module

The flow table module maintains all of the current active network flow information. Once a packet is received, it checks whether the packet belongs to an existing network flow. If not, a new flow record is created in the flow table and contains flow characteristics and some statistical information, such as the total number of bytes processed, the start time and the arrival time of the last packet of the flow. Due to the limitation of main memory size in practice, some expired or stale flow records need to be deleted from the flow table to maintain a suitable flow table size.

Index module

The index module will write each incoming packet to a file and generate the index information of the packet simultaneously. The index module maintains all traffic files, and keeps each traffic file within a fixed size. Once the file size reaches a pre-configured number, a new file is created to store the new arriving packets.

Query Module

The query module is responsible for responding to the user's query by parsing the user's query and executing the query operation. It first parses the user's query request into its own query data structure. It then sends this data structure to the query processing engine, which searches all of the indexes to find the matched index. Then, the query module collects all of the files with matched indexes, extracts the corresponding packets and merges them into a single file which will be returned to the user.

c) TIFAflow deployment

TIFAflow implementation is running on commodity Intel G41 based platform [34] to handle 1 Gbps-10 Gbps network traffic with Quad Core QX9400 or E3125.

CNSMS [35-40] consists of tens of TIFAflow systems which have been widely deployed as a collaborative security overlay network in real network environments. CNSMS also consists of a lot of well deployed UTM node named NetSecu.

The backend of CNSMS is a security center [37,38] based on cloud computing. CNSMSs cloud storage is based on hadoop distributed file system with 40 physical servers, which accumulated volume is about 40TB.

Luca Deri et al. [41] propose a distributed architecture that adopt a small-sized cloud to provide a consistent data space for traffic archival. Their cloud node uses Redis key value storage.

4 Performance Evaluation

4.1 Experimental settings

a) Data Source

TraceA, hereafter the name of experimental data, was collected with TIFA from the IDC operated by Beijing Capital Info Company.

The traffic anonymization and content analysis based on cloud computing are also described in [42]. In this evaluation, we extract the TraceA data set with a total size about 102 GB.

Two sets of queries operations were constructed, and were named Q1 and Q2. Each query set has 10 different query statements. Q1 contains all the global query operations without time intervals. The time interval of the query operation is less than 600 seconds or 10 minutes in the Q2 set.

b) Index Field for a Packet

A packet triggers an index, and the index fields are shown in Table 4. Each index contains five-tuple information, a time stamp, and is stored in a packet. The separation of each byte in an IP address will make queries more flexible.

4.2 MySQL scheme

In this paper, we used the relational database MySQL database version 5.1. According to the definition of index, we used the database table, whose format is defined in Table 5. Here the timestamp is chosen as an index column, and we will discuss the variation in the querying performance when indexing the timestamp column and without indexing any columns.

Table 4 Packet index format

Field	Type	Description
sip1	byte	The 1st byte of Source IP Address
sip2	byte	The 2nd byte of Source IP Address
sip3	byte	The 3rd byte of Source IP Address
sip4	byte	The 4th byte of Source IP Address
dip1	byte	The 1st byte of Destination IP Address
dip2	byte	The 2nd byte of Destination IP Address
dip3	byte	The 3rd byte of Destination IP Address
dip4	byte	The 4th byte of Destination IP Address
sport	short	Source Port
dport	short	Destination Port
time	double	Time Stamp
fileno	int	pcap File Number
offset	int	Offset in pcap file
protocol	int	Transfer protocolTCP/UDP

Table 5 The table structure stored in the MySQL database

Index Design		MySQL Table Design		
field	type	column	type	Y/N
sip1	byte	sip1	TINYINT UNSIGNEDNOT NULL	No
sip2	byte	sip2	TINYINT UNSIGNEDNOT NULL	No
sip3	byte	sip3	TINYINT UNSIGNEDNOT NULL	No
sip4	byte	sip4	TINYINT UNSIGNEDNOT NULL	No
dip1	byte	dip1	TINYINT UNSIGNEDNOT NULL	No
dip2	byte	dip2	TINYINT UNSIGNEDNOT NULL	No
dip3	byte	dip3	TINYINT UNSIGNEDNOT NULL	No
dip4	byte	dip4	TINYINT UNSIGNEDNOT NULL	No
sport	short	sport	SMALLINT UNSIGNED NOT NULL	No
dport	short	dport	SMALLINT UNSIGNED NOT NULL	No
time	double	time	DOUBLE NOT NULL	No
fileno	int	fileno	MEDIUMINT NOT NULL	No
offset	int	offset	MEDIUMINT NOT NULL	No
protocol	int	proto	MEDIUMINT NOT NULL	No

4.3 MySQL vs. TM

Two MySQL schemes, i.e., without building the index for any column and building the index for only timestamps column, are evaluated with TraceA. Similarly, both query set Q1 and Q2 are still used to measure the efficiency of the query. The system memory consumption the time spent building the index, and the query efficiency are determined from the experiments, and the results are shown in Figures 7 and 8.

In Figure 7, the scheme using MySQL to store packet index information is seen to have a poorer performance compared with the TM scheme. Without building the index for any column, the storage processing time is about 4 times the value in TM. When building the index for the timestamp column, the storage processing requires 50% more time than that with no indexes built for any column.

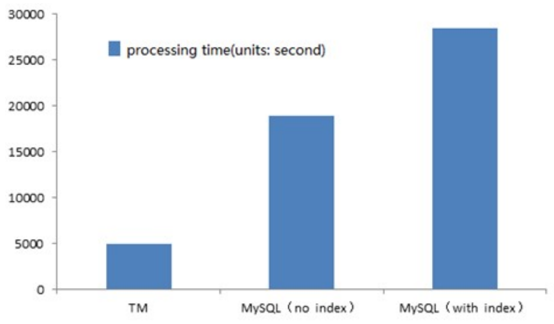


Fig. 6 Comparison of storage performance for MySQL and TM based schemes

Figure 8, it shows that the scheme used to store packet index information with MySQL database has poorer query performance, and a query takes a much longer time than does TM. In addition, it is contrary that the query spends more time both for query sets {Q1, Q2}, even after building the index for the timestamp column. This shows that the timestamp column is not suitable for indexing to speed up query operations. For the TM, executing a Q2 query is faster than executing a Q1 query because the size of the searched space is also reduced when the given time interval is narrowed.

For the scheme using MySQL to store packet index information, a query in the Q2 set takes a longer time than a query in the Q1 set, whether or not an index is built for each column. This is because the addition of a new query condition to the statement results in greater time consumption in each query.

By performing the above performance evaluation,

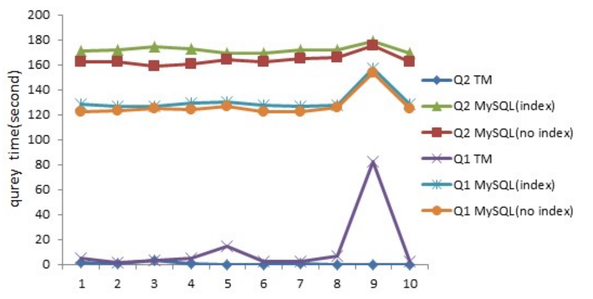


Fig. 7 Querying the performance comparison of MySQL and TM based schemes

the following conclusion can be reached: a common relational database is not suitable for direct use in storing the packet index information.

4.4 TM vs. TIFA FastBit

TraceA and query sets {Q1, Q2} are also used to evaluate the processing and query performance of the system implemented with TIFA FastBit database (hereinafter referred to as the FastBit scheme). The experimental results are shown in Figures 9 and 10.

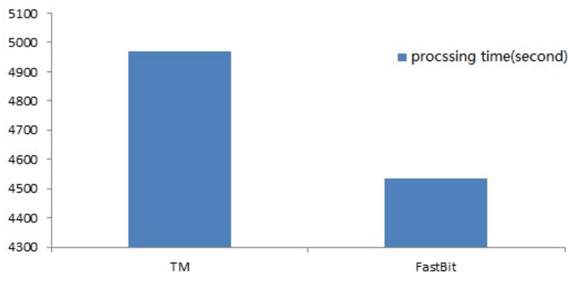


Fig. 8 Storage performance comparison between TIFA FastBit and TM schemes.

With respect to storage performance, the FastBit scheme is about 10% faster than the TM scheme, and establishes an index for each packet. The TM scheme will drop a number of indexes when the processing speed cannot keep pace with the packet arrival rate. The storage performance of the TIFA FastBit scheme is higher because it stores the packet index in a bitmap database and uses a bulk data storage mechanism.

For the query performance, the TIFAs FastBit scheme also has a higher efficiency. For query sets { Q1, Q2 }, the TIFA FastBit scheme is faster than the TM scheme, especially for the 9th point in Q1 and Q2 sets in Figure 10. This result shows some differences from the other points. More time is spent at the 9th point compared to others in the Q1 set. As the query result of the 9th point is relatively large (the generated query result is about 23MB), more time is therefore spent. In the Q2 set, there is no matching record in the storage file with the intended input. In this case, the worst case is evaluated. If there are no flow records in a specified time interval, the TM scheme will stop seeking. This is the reason why the TM scheme is faster than the TIFA FastBit scheme. The experimental results show that the TIFA FastBit scheme has a higher efficiency in terms of both the query performance and processing time when used to store index information.

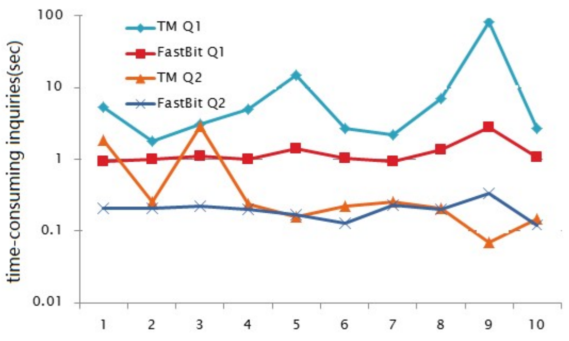


Fig. 9 Comparison of querying performance of FastBit and TM based schemes

5 TIFAflow

5.1 Bottleneck analysis of TIFA

The time taken to build the index and query response is greatly reduced in the TIFA FastBit scheme. However, the index files are relatively larger than before. In the previous scheme, each packet has an uncompressed 32-byte index. Assuming an average packet length of 300 bytes, the additional storage consumption of the indexes is about 10%. On the other hand, this scheme have the same number of the index and packets, which is the bottleneck that occurs when storing and querying as the redundancy.

5.2 Index for a flow

To reduce the number of indexes while not losing high efficiency when extracting the packets that match the query statement, another approach is needed to store this information. An ideal approach is to store the packets in the same flow sequentially instead of based on the timestamp; thus, we can combine the information of several indexes in one index. In addition, with the reduction in the number of indexes, the index information, which has been condensed will be represented by the relative position information of packets. Packets in the same flow are stored together, with the index marking the file location of the flow.

Therefore, this approach will speed up the efficiency of indexing and reduce the required storage space. The format of the index of a flow is defined in Table 6.

Table 6 Index format of a flow

Field	Type	Description
sip1	byte	The 1st byte of Source IP Address
sip2	byte	The 2nd byte of Source IP Address
sip3	byte	The 3rd byte of Source IP Address
sip4	byte	The 4th byte of Source IP Address
dip1	byte	The 1st byte of Destination IP Address
dip2	byte	The 2nd byte of Destination IP Address
dip3	byte	The 3rd byte of Destination IP Address
dip4	byte	The 4th byte of Destination IP Address
sport	short	Source Port
dport	short	Destination Port
time_start	double	Flow beginning timestamp
time_end	double	Flow ending timestamp
file_no	int	pcap File Number
offset	int	offset in pcap file
flowlen	int	volume of flow occupied file
protocol	int	Transfer protocolTCP/UDP

This scheme will also be very convenient for searching all of the packets in the same flow, and will not impair system performance. By using the above approach, if a flow contains n packets, these packets will need only one index, significantly reducing the number of indexes that needs to be stored, hence easing the burden on the database.

5.3 TIFAflow design

TIFAflow implement storage on flow granularity, the network flows are cached in memory, where packets in one flow are stored together. A linked list is inserted into the connection object in the existing flow table. Each node in the linked list is a packet. The detailed process is as follows: after capturing a packet, the system does not store this packet immediately, but adds it to the end of the linked list which has the same flow information (discard if it is out of the truncated boundaries). When the system deletes the time-out flow information in the flow table, it stores the linked list packets of the flow information into the pcap file, and generates an index into the database.

During the querying operation, the index is searched. When matched, the matched target index will be extracted to retrieve the flow in a pcap file, the offset bytes, and the number of bytes. The entire flow data will be immediately retrieved in the appropriate file location.

5.4 TIFAflow vs. TIFA Fastbit vs. TM

We evaluated the storage and indexing time required for TIFAflow with TraceA, and made comparisons with the time required by the TM scheme and the TIFA FastBit schemes, as shown in Figure 11. It is obvious that using the flow granularity storage scheme improves the performance of the whole system. The time taken is reduced by 752 seconds compared with the schemes, which do not use the flow granularity storage. That is, TIFAflow scheme is about 16.6% faster than TIFA Fastbit.

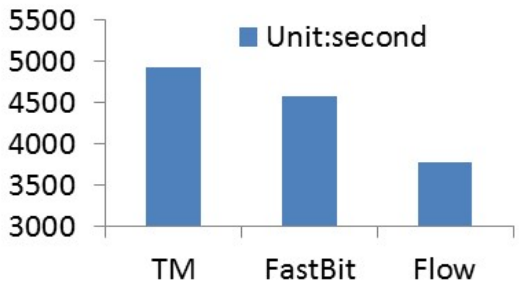


Fig. 10 Querying performance of the TM, TIFA and TIFAflow.

Comparing the size of the generated index files in Figure 12, TIFAflow uses the flow level storage with the TIFA FastBit indexing scheme, which reduces the entire storage size by a factor of six.

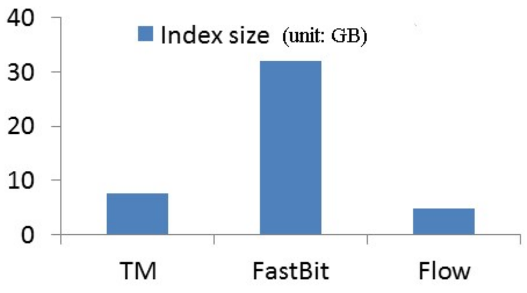


Fig. 11 The size of indexes of TM, TIFA and TIFAflow.

In all, based on flow granularity storage and index scheme, TIFAflow improves the storage and indexing efficiency, while reducing the index file size significantly.

6 Conclusion

Based on the well-known TimeMachine, new design ideas and implementations were explored for Internet traffic archiving and querying system based on flow granularity, and we rewrite a flow based indexing, storage and query module in TimeMachine. Several sets of experiments were conducted to evaluate the scheme based on the relational database MySQL and the FastBit bitmap index of the database, and we evaluated the storage and query performance. Experimental results show that this novel approach can reduce the cost associated with the storage of indexes, and can significantly speed up the flow querying operation, Further, we combined the proposed flow granularity storage with FastBit indexing schemes. Experimental results show that it can further reduce indexes storage and speed up the query operation. In addition, the experiments show that the system is workable in 1 Gbps10 Gbps high-speed network environments for the archiving and querying of network traffic with commodity hardware.

Acknowledgements

This work is supported by Ministry of Science and Technology of China under National 973 Basic Research Program Grant No.2012CB315801, Grant No.2011CB302805, China NSFC A3 Program (No.61161140320) and NSFC No.61233016.

This work is also support with Intel Research Councils UPO program with title of Security Vulnerability Analysis based on Cloud Platform with Intel IA Architecture.

We also thanks for Shuai Ding, Fuye Han, Jun Li, Yong Liang, Er-Long Min and Xi Shi for their work in NSlab-Saturn team.

References

[1] B. Wardman, G. Shukla, and G. Warner, "Identifying vulnerable websites by analysis of common strings in phishing URLs." Published in eCrime Researchers Summit, 2009. eCRIME '09, Tacoma, WA, 2009.

[2] S. Li, R. Schmitz, A novel anti-phishing framework based on honeypots. Published in eCrime Researchers Summit, 2009. eCRIME '09, pp. 1-13.

[3] R. Layton, P. Watters, R.Dazeley, Automatically determining phishing campaigns using the USCAP methodology, Published in eCrime Researchers Summit (eCrime), 2010. pp. 1-8.

[4] S. Sheng, B. Wardman, G. Warner, L. Cranor, J. Hong, C. Zhang, An Empirical Analysis of Phishing Blacklists, in Proc. Sixth Conference on Email and AntiSpam (CEAS 2009), July 16-17, Mountain View, California, USA, 2009.

[5] Zhen Chen and Chuang Lin et al., AntiWorm NPU-based parallel bloom filters for TCP/IP content processing in Giga-Ethernet LAN, Local Computer Networks, 2005. 30th Anniversary, Sydney, NSW, pp. 748-755.

[6] Donghua Ruan and Zhen Chen et al., Handling High Speed Traffic Measurement Using Network Processors, Communication Technology, 2006. ICCT '06. Guilin, pp. 1-5.

[7] Shihai Huang and Zhen Chen et al., Proxy-based Security Audit System for Remote Desktop Access, Computer Communications and Networks, 2009. ICCCN 2009. San Francisco, CA, pp. 1095-2055

[8] C. Cranor, T. Johnson, and O. Spatscheck. Gigascope: a stream database for network applications. /textit2003 ACM SIGMOD international conference on Management of data, NY, USA, pp. 647-651

[9] M. Sullivan and A. Heybey. Tribeca: A system for managing large databases of network traffic. /textitProceedings of the USENIX Annual Technical Conference (NO 98), ew Orleans, Louisiana, June 1998

[10] L. Deri, V. Lorenzetti, and S. Mortimer, Collection and exploration of large data monitoring sets using bitmap databases, in *Traffic Monitoring and Analysis(TMA)*, Jan 2010. pp. 73-86.

[11] P. Desnoyers and P. Shenoy, Hyperion: High Volume Stream Archival for Retrospective Querying, USENIX Annual Technical Conference, Santa Clara, CA, 2007.

[12] Stefan Kornexl, Vern Paxson, Holger Dreger, Anja Feldmann, Robin Sommer, Building a Time Machine for Efficient Recording and Retrieval of High-Volume Network Traffic, *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, CA, USA, pp. 23

[13] G. Maier, R. Sommer, H. Dreger, A. Feldmann, V. Paxson, and F. Schneider. Enriching Network Security Analysis with Time Travel. *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, NY, USA, pp. 183-194.

[14] L. Deri, A. Cardigliano, F. Fusco, 10 Gbit Line Rate Packet-to-Disk Using n2disk, Proc. of 2013 Traffic Monitoring and Analysis workshop, TMA 2013.

[15] TimeMachine project in Bro, see http://tracker.bro.org/time-machine/.

[16] Jun Li, Shuai Ding , Ming Xu , Fuye Han, Xin Guan and Zhen Chen. TIFA: Enabling Real-Time Querying and Storage of Massive Stream Data. Networking and Distributed Computing (ICNDC), 2011 Second International Conference on IEEE, Beijing, 2011

[17] K. Wu and others, FastBit: Interactively Searching Massive Data, In *Proc. of SciDAC 2009*, 2009.

[18] B. Claise, Cisco Systems NetFlow Services Export Version 9, RFC 3954, 2004.

[19] P. Phaal, S. Panchen, N. McKee, InMon Corporations sFlow: A Method for Monitoring Traffic in Switched and Routed networks, RFC 3176, 2001.pp. 1-31.

[20] P. Haag, Watch your Flows with NfSen and NfDump, 50th RIPE Meeting,Stockholm, Sweden, 2005.

[21] M. Fullmer and S. Roming, The OSU Flowtools Packetage and Cisco NetFlow Logs, /textitIn Proc. Of 19th Intl. Conference on Scientific and Statistical Database Management, Banff, Canada, 2007.

[22] NEye, an Open Source NetFlow collector, http://neye.unsupported.info, 2004.

[23] J. P. Navarro, B. Nickless, L. Winkler, Combining Cisco NetFlow Exports with Relational Database Technology for Usage Statistics, Intrusion Detection, and Network Forensics, *Proceedings of the 14th Large Installation Systems Administration Conference (LISA 2000).* 2000. pp. 285-290.

[24] F. Fusco, X. Dimitropoulos, M. Vlachos. pcapIndex: an index for network packet traces with legacy compatibility. ACM SIGCOMM Computer Communication Review, vol.42, no.1, pp. 47-53. January 2012

[25] F. Fusco, M. Vlachos, X. Dimitropoulos, L. Deri, Indexing million of packets per second using GPUs, Proc. of the 13th ACM SIGCOMM conference on Internet Measurement Conference, IMC 2013.

[26] F. Fusco and L. Deri, High speed network traffic analysis with commodity multi-core systems, *of the 10th ACM SIGCOMM conference on Internet measurement*, NY, USA, pp. 218-224.

[27] Cavium Network Processor OCTEON 58XX,see http://www.cavium.com/OCTEON-Plus_CN58XX.html.

[28] J. Meng, X. Chen, Z. Chen, C. Lin, B. Mu, L. Ruan. Towards High-performance IPsec on Cavium OCTEON Platform. in *Trusted Systems*, 2011, pp. 37-46.

[29] Tilera manycore Network Processor, see http://www.tilera.com/products/processors/TILEPRO64.

[30] S. Ding, Z. Chen and Z. Liu, Parallelizing FIB Lookup in Content Centric Networking, Networking and Distributed Computing (ICNDC). Hangzhou, China, 2012, pp. 6-10.

[31] V. Paxson, Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks*, vol. 31, no. 23, pp. 2435-2463, 1998.

[32] F. Fusco, M. P Stoecklin, and M. Vlachos. "NET-FLi: on-the-fly compression, archiving and indexing of streaming network traffic." Proceedings of the VLDB Endowment vol.3, no. 1-2 pp. 1382-1393. 2010

[33] Z. Chen et al. High speed traffic archiving system for flow granularity storage and querying. Computer Communications and Networks (ICCCN), Munich, 2012, pp. 1-5.

[34] X. Chen, B. Mu, Z. Chen, NetSecu: A collaborative network security platform for in-network security, in *Proc. the 3rd International Conference on Communications and Mobile Computing (CMC)*, Qingdao, China, 2011, pp. 59-64.

[35] Beipeng Mu, Xinming Chen and Zhen Chen. A Collaborative Network Security Management System in Metropolitan Area Network. Communications and Mobile Computing (CMC), 2011 Third International Conference, Qingdao, 2011, pp. 45-50.

[36] Deng, Fachao, et al. TNC-UTM: A holistic solution to secure enterprise networks. Young Computer Scientists, 2008. ICYCS 2008. Hunan, 2008, pp. 2240-2245

[37] Z. Chen et al. Cloud computing-based forensic analysis for collaborative network security management system. Tsinghua Science and Technology vol.18, no.1, pp. 40-50. 2013

[38] F. Han, Z. Chen, H. Xu and Y. Liang. Garlic: A Distributed Botnets Suppression System. Distributed Computing Systems Workshops (ICDCSW), Macau, 2012, pp. 634-639.

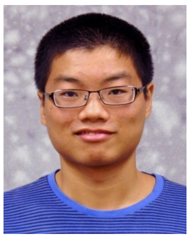
[39] A. Luo, L. Chuang, Z. Chen, X. Peng, and P. D. Ungsunan. TNC-compatible NAC System implemented on Network Processor. Local Computer Networks, Dublin, 2007. pp. 1096-1075.

[40] Z. Ying, F. Deng, Z. Chen, Y. Xue, and C. Lin. UTM-CM: A practical control mechanism solution for UTM system. Communications and Mobile Computing (CMC), Shenzhen, 2010, pp. 86-90

[41] L. Deri, F. Fusco, MicroCloud-based Network Traffic Monitoring, Proc. of the Intern. Symposium on Integrated Network Management, IM 2013.

[42] Tianyang Li, Fuye Han, Shuai Ding and Zhen Chen. LARX: Large-scale Anti-phishing by Retrospective Data-Exploring Based on a Cloud Computing Platform. *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on IEEE.*, Maui, HI, pp. 1-5.

Zhen Chen is an associate professor in Research Institute of Information Technology at Tsinghua University. He received his B.E. and Ph.D. degrees from Xidian University in 1998 and 2004. He works as postdoctoral researcher leaded by Prof. Chuang Lin in Network Institute of Department of Computer Science in Tsinghua University during 2004 to 2006,. He is also a visiting scholar in network group of ICSI in UC Berkeley, which leaded by Prof. Scott Shenker in 2006. His research interests include overlay networking architecture, Internet security and Data analysis. He has published around 80 academic papers. Now, he is leading the nslab-saturn team (www.nslab-saturn.net).



Lingyun Ruan is a Ph.D. student from Department of Computer Science in Purdue University. He worked in network secueity lab of RIIT and finished his research on UTM in 2011. He got his BEng degree from Tsinghua University in 2011. Currently his main research interests focus on data mining and machine learning.

Junwei Cao is currently Professor and Deputy Director of Research Institute of Information Technology, Tsinghua University, China. He is also Director of Open Platform and Technology Division, Tsinghua National Laboratory for Information Science and Technology. His research is focused on advanced computing technology and applications. Before joining Tsinghua in 2006, Junwei Cao was a Research Scientist of Massachusetts Institute of Technology, USA. Before that he worked as a research staff member of NEC Europe Ltd., Germany. Junwei Cao got his PhD in computer science from University of Warwick, UK, in 2001. He got his master and bachelor degrees from Tsinghua University in 1998 and 1996, respectively. Junwei Cao has published over 130 academic papers and books, cited by international researchers for over 3000 times. Junwei Cao is a Senior Member of the IEEE Computer Society and a Member of the ACM and CCF.

Yifan Yu is an undergraduate student working in Department of Electronic Engineering at Tsinghua University. His research interests include network security and mobile safety.

Xin Jiang is working as computer security researcher. He received the Ph.D. degree in Computer Science from Institute of computer network of Department of Computer Science in Tsinghua University in 2010. He got bachelor's degree in PLA University of science and technology in 1998. His main research interests include computer network security, performance evaluation and wireless networks.