Building a Feedback Loop to Capture Evidence of Network Incidents

Zdeněk Rosa[†], Tomáš Čejka[*], Martin Žádník[*] and Viktor Puš[*]
[*]CESNET, a.l.e. [†]CTU in Prague, FIT
Zikova 4 Thákurova 9
160 00 Prague 6, Czech Republic
{zadnik,cejkat,pus}@cesnet.cz, rosazden@fit.cvut.cz

***Abstract*—Flow measurement is extremely useful in network management, however, in some cases it is vital to observe the packets in full detail. To this end, we propose combining flow measurement, packet capture and network behavioral analysis. The evaluation of the proposed system shows its feasibility even in high-speed network environment.**

I. Introduction

Flow-based network traffic monitoring, that is, aggregating defined packet-header fields into the flow records the packets belong to, is crucial for an abstract yet detailed network visibility. Especially on high speed links, there is a need to reduce the amount of monitored data into aggregated information to enable on-the-fly analysis and continuous storage for a long period of time offering administrators an overview of what has happened in the network in the past. Inevitably the data reduction leads to an information loss. However, for certain applications (e.g. security forensics, network trouble-shooting) it is useful to capture the traffic of interest in full detail. For example, in case of an attack, a sample of the packet data can be used to derive its unique pattern leading to its blocking (like in [1]). Packet data can be used for validation of the compromise phase e.g. for VoIP fraud [2] or password attack.

Rather than to propose a new approach to network monitoring we have decided to take an incremental path and amend flow-based monitoring with automated full packet capture to provide an evidence of suspicious events. The system combines several concepts ranging from Software Defined Monitoring (SDM) [3], [4], the Time-machine [5], the flow-monitoring with network behavioral analysis [6]. SDM controls hardware acceleration of the measurement. The Time-machine provides functionality to access full packet data even from the time before the event of the interest was detected. The flow-monitoring with behavioral analysis detects an event of interest and provides a feedback-loop for selective packet capture of related packets.

The rest of the paper is structured as follows. Sec. II discusses related work. Sec. III proposes the monitoring architecture and we evaluate characteristics of the architecture in Sec IV. Sec. V concludes the paper.

II. Related Work

Although it is possible to capture all the traffic even on a high speed link [7] the enormous amount of data renders it infeasible for a long-term storage. Therefore, various aggregation approaches appeared, among others IP flow monitoring represented by several generations of NetFlow protocols [8] and IPFIX [9]. Various optimizations have been proposed to simplify flow measurement either by focusing on a large flows such as in [10], [11] or by estimating characteristics from sampled traffic directly [12]. These optimizations benefit from the inherent heavy-tailed behavior of certain network traffic characteristics [13], e.g. a small number of flows accounts for a large portion of the traffic.

SDM benefits from the heavy-tailed distribution of the flows as well. The core idea of SDM is to analyze and measure first N packets of each flow in software and then to decide whether to offload or not the measurement of the rest of the flow into a dedicated network card such as [14]. The primary utilization of SDM is an application-aware flow measurement as the application information is extracted from the start of the flow and included into the exported flow records.

The Time-machine [5] proposes to store first N packets in two ring buffers allocated in RAM and on the disk where the RAM allows for faster access to the traffic captured recently. Since there is no pruning of the captured traffic when it is transferred from RAM to the disk, the disk may become a bottleneck of such a system in the high-speed environments either from the point of view of the storage speed[1] or the capacity.

COFFEE [15] tries to do a similar task using SDN, however, there are many differences such as level of detail, performance requirements (10,000 flows/s, we need at least 150,000 flow/s).

Our work tries to combine the best of the above described approaches. We motivate our work by these assumptions:

- flow measurement could be simplified but only for given applications, however, in real-life deployment the utilization of flow measurement is many-fold and ad-hoc, hence, the full flow measurement is vital,
- the start of a flow is important, especially for security and troubleshooting,
- there is a need to swiftly select interesting traffic from the history stored in the ring-buffer.

[1]According to our experience, SSD disks are more prone to errors in case of these types of tasks, i.e. overwriting the whole disk capacity periodically

978-3-901882-85-2 © 2016 IFIP

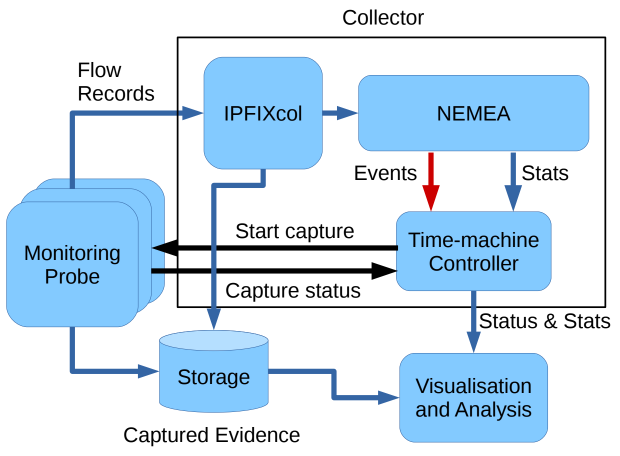

Fig. 1. High level architecture of SDM infrastructure using feedback from the NEMEA detection system.

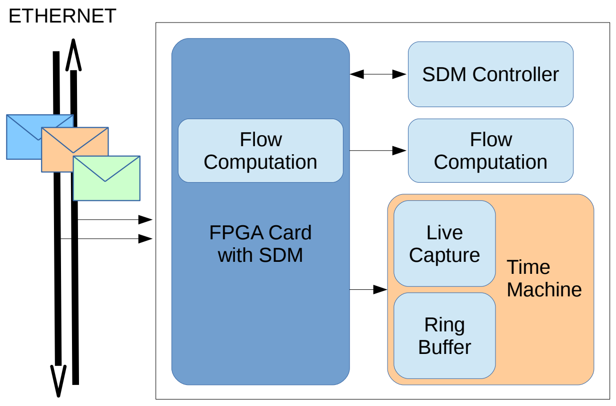

Fig. 2. SDM probe extended by Time-machine.

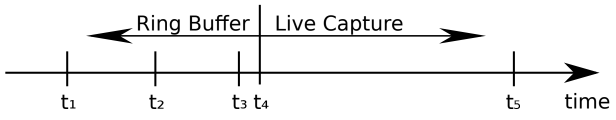

Fig. 3. Timeline of the Time-machine

III. Design and Architecture

The whole monitoring system is depicted in Fig. 1. Compared to a classical flow-based architecture [16], it is extended. It utilizes SDM capable monitoring probe to measure and export network flows to a collector. The collector employs several tools to perform multiple tasks. The flow processing is implemented by IPFIXcol [17] that streams the flow records into NEMEA [6]. NEMEA is a stream-based network behavioral engine containing various detection modules. It provides alerts and statistics to the Time-machine Controller. The controller converts the alerts into the capture rules implementing the feedback for the probes to select traffic from the Time-machine, which is inside a monitoring probe, and to trigger selective traffic capture at the probe. The rules are sent directly to the Time-machine using secured communication channel. Subsequently, the alerts, statistics, flow records and the evidence traffic are visualized together to simplify the analysis and allow for the system management.

Monitoring probe with SDM concept is based on a co-design of hardware-accelerated network card and specific software components such as SDM controller, flow measurement and, in our case, packet capture. The controller is aware of statistics on incoming network traffic as well as of the requests for delivering particular piece of traffic to the other software components. The controller decides accordingly when and which flows should be offloaded to utilize hardware resources efficiently. As a result, partial statistics are computed both in software and hardware (Flow Computation blocks in Fig. 2) and the flow records are sum of these statistics.

In order to retrieve additional information we implemented the packet capture in the probe. We call the packet capture functionality the Time-machine according to the work [5] it was proposed in, but our Time-machine differs to the original concept. It does not utilize the disk buffer and it captures all the packets of the suspicious flows once the event was detected.

Our Time-machine consists of two components shown in Fig. 2. *Ring Buffer* provides beginnings of flows from the time prior to the reported alert and *Live Capture* provides full packet capture since the alert.

Fig. 3 shows points in time that are related to a single detected event. The event starts in t_1 and lasts until t_5. Packets of the event are being exported as flow records after t_2. The event is successfully detected in t_3, where $t_3 - t_1$ gives us a detection delay that is caused by exporting delay $t_2 - t_1$ and the nature of the utilized detection algorithm. The exporting delay is primarily caused by the timeouts of the flow measurement and less by the transfer and related buffering. The delay between t_3 and t_4 is is in order of milliseconds, and is caused by Ethernet round-trip-time. It is not significant contrary to other delays. The output of both Ring Buffer and Live Capture is a set of PCAP files containing full packets.

Ring Buffer stores the first N packets of each flow into memory. After detection, Ring Buffer looks up all packets of the suspicious IP and stores them into a file on the disk. The size of Ring Buffer is given by the size of the allocated memory. The size of stored history depends on the Ring Buffer size, N and current traffic distribution. Live Capture stores all packets of the selected IP into a file on the disk directly.

The whole architecture was described using an example with a single monitoring probe. In practice, there are usually many observation points where an operator needs to monitor the network (Fig. 1 shows multiple Monitoring probes). Our monitoring infrastructure takes into account such a distributed environment. Following the best practice of flow measurement, the captured packets of interest are collected to the single collector via TLS connection. The Time-machine controller is able to control the capture trigger at each probe individually.

IV. Evaluation

The whole monitoring infrastructure was deployed in CESNET2 [18] network in particular, for the purpose of the

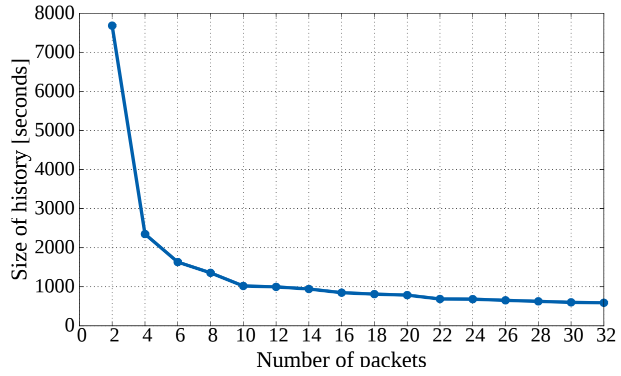

Fig. 4. Length of history kept in the Time-machine ring buffer.

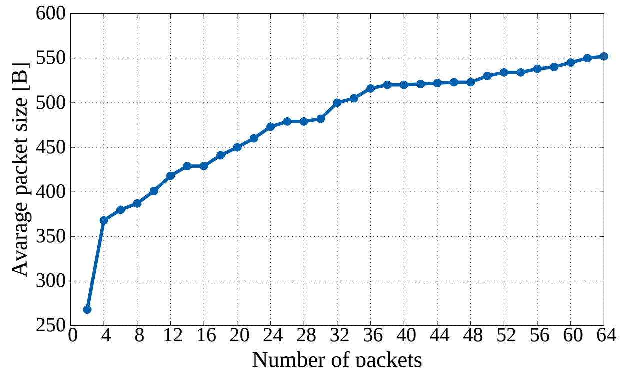

Fig. 5. Average packet size due to the number of packets from flow

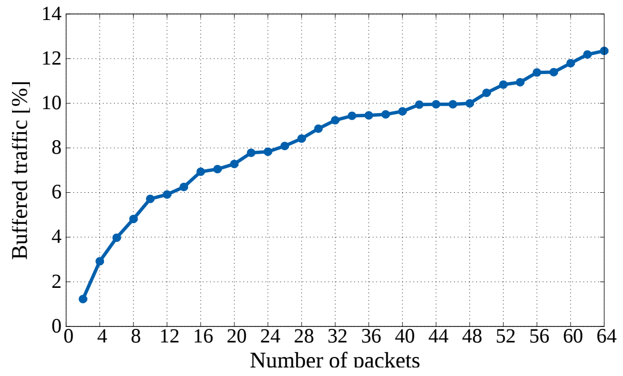

Fig. 6. Percentage of buffered traffic depending on N

evaluation, we work with the link between CESNET2 and ACOnet [19]. The probe monitors both directions with dual port 10 Gbps hardware accelerated network card with SDM enabled. The maximum link utilization during our evaluation was nearly 6 Gbps (800 Kpackets/s, 30 Kflows/s) in each direction while the long-term average is approx. 2 Gbps (250 Kpackets/s, 15 Kflows/s). The probe must primarily measure flows. The flow measurement demands large portion of RAM since the number of concurrent flows is high (nearly 1 mil. of concurrent flows records in the flow cache during peaks). Therefore, we allocate only moderate 8 GB of RAM for the Time-machine ring buffer. With this setup we experimented with various settings of N (the number of the first packets in each flow). The lower the N, the longer history we can hold.

Fig. 4 shows the length of the history it is possible to keep in RAM depending on the N. If we keep only two packets we can capture and hold the packets over two hours. However, if we increase N to four the length drops to one third. Such a decrease is explained by the small sizes of the first two packets in the TCP flow while the rest of the packets are typically larger. Beginning of the flow mostly transfer some negotiation information like establishment of communication. The rest of the flow usually transmits application data and it uses larger sizes of the packets to increase the throughput between communication nodes. Fig. 5 shows average size of the buffered packets with respect to N. On the other hand, we can see in Fig. 4 the larger the N, the less significant decrease. Such a characteristic is expected and reflects the heavy-tailed distribution of flow sizes (see Fig. 7). The number of flows with more than N packets drops dramatically with larger N.

Fig. 7 (*all* line) shows the percentage of total number of flows accounting for less than N packets. The graph shows that choosing 10 as a maximal packet threshold, we can see about 90 % flows that contain less or equal number of 10 packets, i.e. the 90 % flows can fit to 10 packets.

Fig. 6 shows the influence of N on the percentage of buffered traffic. Again, the heavy-tailed distribution of flow sizes plays its role, since the majority of the traffic is transferred in a small number of flows which are large. The larger the flow is the more packets gets discarded.

Percentage of the buffered traffic (Fig. 6) and the packet size (Fig. 5) have the main effect on the length of history that fits in the buffer (Fig. 4). As the minimum, the time-machine must buffer the packet until its corresponding flow is exported to the collector. Such a minimum may be derived from active and inactive timeout of the flow measurement (typically set to 30 s and 300 s respectively, or less). For the 8 GB buffer and throughput of 5 Gbps, the minimal interval (330 s) corresponds to $N = 88$ given the average packet size of 560 B. The actual length of history we need to buffer depends on the use case because we must take into account the time to detect the event, i.e. to receive sufficient number of flow records to recognize that an event has happened. The time spent on conversion and communication is negligible, buffer pruning and live capturing starts in less than 1 s after an alert is reported by a detection module. Therefore, we are most interested in the delay between the first suspicious packet and the detection, the detection delay $t_3 - t_1$. The detection delay is measured as a difference between start timestamp in the first suspicious flow and the time when the alert is reported. The detection time depends on the detection algorithm and its setup individually.

NEMEA employs several modules to detect various types of events. These modules reported 10,598,587 events during one month thereof 9,746,163 unique events (an event can be reported multiple times if it lasts sufficiently long). The detection delay is the lowest in case the flows are exported

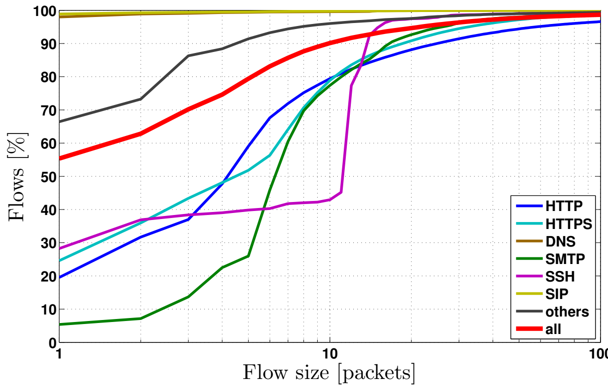

Fig. 7. Percentage of flows shorter than the given flow size

from the probe immediately (e.g. due to FIN flag or full flow cache and not due to the timeouts) and the detection algorithm works upon a single flow (e.g. to check whether an IP address in the flow record is on a blacklist) or limited number of flows (e.g. compute statistics per each host and checks regularly if the host characteristic corresponds to an attack pattern).

Other detectors usually require to observe incoming flows for a longer period, especially, detectors focusing on under-the-threshold attacks (e.g. slow bruteforce attacks or VoIP fraud). These attacks are of low intensity, hence, it takes time for the detector to receive sufficient number of flow records to recognize the pattern. And the lower threshold would only lead to the increase of false positives.

Fig. 8 depicts empirical distribution function of the detection delay in total as well as per each deployed module. The distribution function for the total detection delay shows that in order to keep first packets for 80 % of events we need more than 900 s of history. This roughly corresponds to $N = 10$. However, this is too coarse as the goal to capture the traffic may differ per each module. In case of intensive attacks we need the sample of the attack before the attack ends. Since the majority of intensive attacks is easy to detect early, $N = 10$ seems to be a good option. Moreover, if the attack prevails at the detection time, then the Live Capture collects further samples. In case of an attack of low intensity we are more interested in the continuation and success of the attack and less in its starting phase. Therefore, we can, for example, increase N to 32 packets to enlarge the captured sample in trade for shortening the history as we know that we cannot capture the first attacking flow in more than 70 % of events in case of the VoIP fraud. By increasing the N we may receive larger portion of the suspicious packets from the buffer and the live capture delivers the rest of the ongoing attack. In other words, we should keep history corresponding to the exporting delay at least (i.e. active plus inactive timeout). As a result, if the buffer does not capture the first suspicious flows it definitely captures the flow, and its packets, that leads to launching the detection as this must be the incoming flow that triggers the detection of an event, i.e. that causes the statistics to exceed the thresholds.

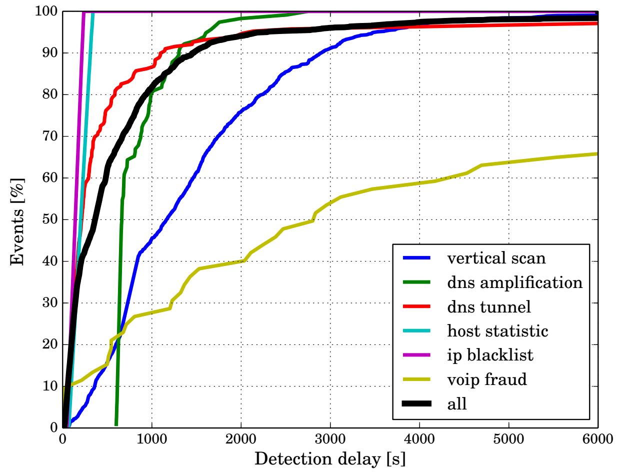

Fig. 8. Detection delay

V. Conclusion

Flow measurement and full packet capture represent complementary approaches to network traffic monitoring. Both are essential for network administration as well as network security. This paper presented an architecture combining advanced flow monitoring with packet capture and the feedback loop from network behavioral analysis engine. The feedback loop allows to select only the traffic of interest either from the packets captured in the past or from the live capture. The evaluation offers various trade-offs to setting N, the number of packets captured from the beginning of each flow in the past. Depending on the target utilization we should select N less than 88 packets but rather lower (10 to 32 seem to be reasonable options) to be able to access at least some suspicious packets in the past.

The proposed architecture has been deployed in CESNET infrastructure where we have utilized it to capture both samples of DNS amplification attacks, communication with C&C server identified by the blacklist and confirmation of VoIP fraud attacks. As our future work we consider to introduce a filter prior to the Time-machine buffer. This filter could discard the portion of the apriori uninteresting traffic (e.g. P2P file transfers) and would lead to prolonging the history.

Acknowledgment

This work was supported by the “CESNET E-Infrastructure” (LM2015042), CTU grant No. SGS16/124/OHK3/1T/18 both funded by the Ministry of Education, Youth and Sports of the Czech Republic and by “The Sharing and analysis of security events in the Czech Republic” project No. VI20162019029 granted by the Security Research Programme of the Czech Republic 2015–2020 Ministry of the Interior of the Czech Republic.

References

[1] C. Kreibich and J. Crowcroft, “Honeycomb: creating intrusion detection signatures using honeypots,” *ACM SIGCOMM computer communication review*, vol. 34, no. 1, pp. 51–56, 2004.

[2] T. Cejka, V. Bartos, L. Truxa, and H. Kubatova, “Using application-aware flow monitoring for sip fraud detection,” in *Intelligent Mechanisms for Network Configuration and Security*. Springer, 2015, pp. 87–99.

[3] L. Kekely, J. Kučera, V. Puš, J. Kořenek, and A. V. Vasilakos, “Software defined monitoring of application protocols,” *IEEE Transactions on Computers*, vol. 65, no. 2, pp. 615–626, Feb 2016.

[4] L. Kekely, V. Pus, and J. Korenek, “Software defined monitoring of application protocols,” in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 1725–1733.

[5] S. Kornexl, V. Paxson, H. Dreger, A. Feldmann, and R. Sommer, “Building a time machine for efficient recording and retrieval of high-volume network traffic,” in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*. USENIX Association, 2005, pp. 23–23.

[6] V. Bartos, M. Zadnik, and T. Cejka, “Nemea: Framework for stream-wise analysis of network traffic,” *CESNET, a. l. e., Tech. Rep*, 2013.

[7] V. Puš, L. Kekely, Špinler Martin, V. Hummel, and J. Palička, “HANIC 100G: Hardware accelerator for 100 Gbps network traffic monitoring,” *CESNET, a.l.e, Tech. Rep*, 2014. [Online]. Available: https://www.cesnet.cz/wp-content/uploads/2015/01/hanic-100g.pdf

[8] B. Claise, “Cisco Systems NetFlow Services Export Version 9,” RFC 3954 (Informational), Internet Engineering Task Force, Oct. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3954.txt

[9] B. Claise, B. Trammell, and P. Aitken, “Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information,” RFC 7011 (INTERNET STANDARD), Internet Engineering Task Force, Sep. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc7011.txt

[10] C. Estan and G. Varghese, “New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice,” *ACM Transactions on Computer Systems (TOCS)*, vol. 21, no. 3, pp. 270–313, 2003.

[11] T. Mori, M. Uchida, R. Kawahara, J. Pan, and S. Goto, “Identifying elephant flows through periodically sampled packets,” in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM, 2004, pp. 115–120.

[12] N. Duffield, C. Lund, and M. Thorup, “Charging from sampled network usage,” in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*. ACM, 2001, pp. 245–256.

[13] K.-c. Lan and J. Heidemann, “A measurement study of correlations of internet flow characteristics,” *Comput. Netw.*, vol. 50, no. 1, pp. 46–62, Jan. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2005.02.008

[14] CESNET, a. l. e., “COMBO card family.” [Online]. Available: https://www.liberouter.org/technologies/cards/

[15] L. Schehlmann and H. Baier, “Coffee: a concept based on openflow to filter and erase events of botnet activity at high-speed nodes.” in *GI-Jahrestagung*, 2013, pp. 2225–2239.

[16] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, “Flow monitoring explained: From packet capture to data analysis with netflow and ipfix,” *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 2037–2064, Fourthquarter 2014.

[17] P. Velan and R. Krejčí, “Flow information storage assessment using ipfixcol,” in *Dependable Networks and Services*. Springer, 2012, pp. 155–158.

[18] “CESNET2 — Czech NREN.” [Online]. Available: https://ces.net

[19] “ACOnet — Austrian NREN.” [Online]. Available: https://aco.net