NEMEA: A Framework for Network Traffic Analysis

Tomas Cejka*, Vaclav Bartos*, Marek Svepes*, Zdenek Rosa* and Hana Kubatova†

*CESNET, a.l.e.
Zikova 4, 160 00 Prague 6, Czech Republic
{cejkat,bartos,svepes,rosa}@cesnet.cz

†CTU in Prague, FIT
Thakurova 9, 160 00 Prague 6, Czech Republic
kubatova@fit.cvut.cz

***Abstract*—Since network attacks become more sophisticated, it is difficult to discover them using traditional analysis tools. For some kinds of attacks, it is necessary to analyze Application Layer (L7) information in order to detect them. However, there is a lack of existing tools capable of L7 processing and manipulation. Therefore, we propose a flow-based modular Network Measurements Analysis (NEMEA) system to overcome the situation. NEMEA is designed with respect to a stream-wise concept, *i. e.* data are analyzed continuously in memory with minimal data storage. NEMEA is developed as an open-source project and is publicly available for world-wide community. It is designed for both experimental and operational use. It is able to process off-line traffic traces as well as live network flows. The system is very flexible and can be easily extended by new modules. The modules are developed within a NEMEA framework that is a key component of the project. NEMEA thus represents a unified platform for research and development of new traffic analysis methods. It covers several important topics not limited to analysis and detection. Originally, NEMEA has been developed for the purposes of Czech National Research and Education Network operator. Therefore, it is focused on handling high speed network traffic with links working at 100 Gbps.**

I. Introduction

Monitoring computer networks belongs to important tasks of every network operator. Monitoring systems can provide valuable information about status and utilization of a network infrastructure. Network security must be kept in mind due to the importance of computer networks, safety of users and their data. Due to the huge volume of data that is transferred via modern network infrastructures, monitoring systems usually aggregate information about traffic into smaller *flow records*. These traditionally consist of network addresses, ports, timestamps and volume information. This data can be used for accounting, statistical analysis to improve situational awareness or for anomaly detection. An overview of concepts of the flow-based network monitoring can be found in [1], [2].

Since network attacks are becoming more sophisticated and hidden, it is sometimes very difficult to recognize them in normal benign traffic. There are several methods for detection of malicious traffic based on traditional flow records. The records can even be used for detection of some attacks on Application Layer (L7) (*e. g.* SSH bruteforce [3]). However, for some kinds of malicious traffic, traditional flow records are not sufficient and information from L7 headers is needed for reliable detection. L7 information is however not well supported in current flow analysis tools.

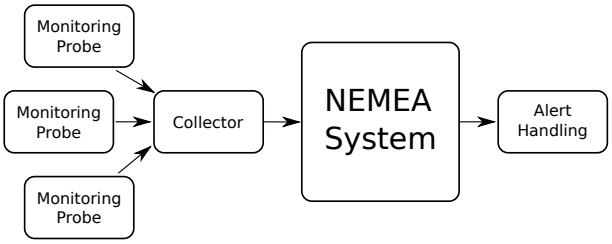

Fig. 1. Monitoring system infrastructure is based on exporting flow records (by Monitoring probes) that are passed for storage (Collector) and analysis (NEMEA). Detection modules of NEMEA produce alerts in a unified format suitable for subsequent processing and storage.

To overcome a lack of existing tools, we developed a new platform for online stream-wise traffic analysis and anomaly detection, capable of L7 data processing – Network Measurements Analysis (NEMEA). Fig. 1 shows a typical flow monitoring infrastructure with traffic analysis done by NEMEA. The network is monitored by monitoring probes (flow exporters) which send flow data to a central collector in form of flow records. The probes contain plugins to parse selected information from L7 protocols and extend the flow records by this information. The IPFIX protocol is used to transfer such data. In our deployment at CESNET2 network (Czech NREN), we use FlowMon [4] probes based on special devices with hardware acceleration in order to process backbone traffic (up to 100 Gbps) without sampling. However, any exporter capable of parsing L7 protocols can be used. Collectors usually store received flow records and in our case the collector (IPFIXcol [5]) also resends them for analysis to the NEMEA system. The probes with the collector are the source of data for analysis. The results of the analysis are statistics of traffic and alerts produced by various detection mechanisms.

NEMEA was firstly introduced in our technical report [6] in 2013 but since that time, it is still being improved. This paper presents main features of NEMEA. The system is composed of independent interconnected modules, it is extensible and it can run in distributed environment. Every module has its own task that can be, for example, anomaly detection, filtering or statistics computation. Each module is built using a set of libraries that create a common framework. Everything is developed as an open-source project and is available at github.com[1].

[1]https://github.com/CESNET/NEMEA

978-3-901882-85-2

This paper is organized as follows. Section II compares the NEMEA system with existing related projects. Section III describes architecture and shows main features of the NEMEA system. Section IV lists real use-cases that we target and Section V presents results that we have achieved using the NEMEA system. Finally, Section VI concludes the paper.

II. Related Work

This section describes related existing systems for traffic monitoring and analysis. The analysis and anomaly detection is often done using Intrusion Detection Systems (IDS) or Intrusion Prevention Systems (IPS). Such systems analyze the network based on packets or network flows. The most popular packet based systems are Bro [7] and Snort [8]. They process every packet and use pattern matching, possibly enhanced by scripting, to search for suspicious traffic and perform actions when a predefined rule matches. Since these systems operate on packets, they can see more details than flow based systems. In contrary, flow based systems usually analyze data on a higher level of abstraction and are thus able to detect different kinds of events. Also, since they process less detailed data, they usually have better performance. These two approaches may complement each other in many scenarios.

Nfdump [9] is a set of tools for storage and processing of flow records. It receives data from the network and stores them into files corresponding to time windows, typically 5 minutes long. Nfsen [10] is a graphical frontend for nfdump that visualizes the stored data in form of graphs and reports. It also allows manual analysis of the data and it can be configured to automatically generate alerts based on simple rules. More advanced data analysis can be done using plugins. The disadvantage of this approach is that the data must be stored to a disk and then read by all the plugins. This is often a performance bottleneck in large networks. Moreover, nfdump does not support flow records extended by L7 information.

The Network Situational Awareness (NetSA) group at CERT created Analysis Pipeline [11] based on SiLK [12]. SiLK is a set of tools for manipulation of records in a flexible data format containing information from flow records. Analysis Pipeline processes the data according to a configuration file that describes a sequence of operations in three stages — filtering, evaluation or statistics computation, and alerting. There is a set of predefined options for each stage. By combining them into a pipeline a complex query can be assembled. Building a complex analysis task from simpler modules is an approach very similar to that of NEMEA. The latest version of Analysis Pipeline also adds a support for L7 data. However, functionality of the building blocks of the pipeline and their possible interconnections are very limited in comparison to NEMEA.

III. NEMEA System

A. Overview

The NEMEA system is designed as a heterogeneous modular system. Modules are independent processes interconnected by unidirectional interfaces for communication. The interfaces transfer data in the form of streams of messages — flow records, results of some analysis etc. A simple example of an instance of the NEMEA system is shown on Fig. 2. Each module is basically a program which performs a specific task, such as flow data preprocessing, filtration, anomaly detection, or logging and reporting results.

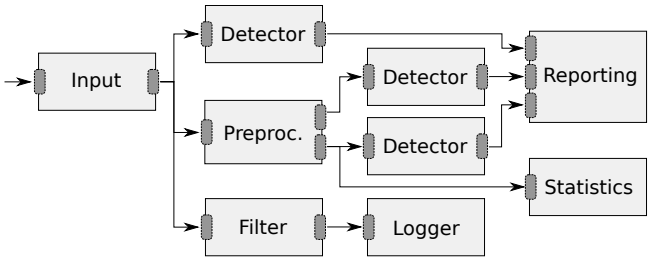

Fig. 2. Example of several NEMEA modules and their interconnection.

Each instance of the NEMEA system can be put together out of different sets of modules, interconnected in various ways. Different deployments of the NEMEA system may use completely different configurations of modules and thus perform different tasks. Therefore, NEMEA is very flexible. In a typical configuration, modules are interconnected into a tree or directed acyclic graph with a single module acting as main input of the whole system. This module gathers or creates flow records and sends them to other modules which process them. On the other end of the system, there are usually modules for logging the results to log files, a database or for sending alerts via email. In our deployment, we use a plugin for IPFIXcol collector as an input of the system — the collector receives and parses IPFIX data from our monitoring probes, the plugin transforms them to a format used by NEMEA and forwards them to NEMEA modules. This was shown in Fig. 1. However, for smaller deployments or for testing it is possible to use NEMEA module *flow meter* as an input. It can read packets from a network interface or a PCAP file, generate flow records and send them directly to the rest of the NEMEA system, so no external exporters and collector are needed.

NEMEA is not only easily reconfigurable, it can also be easily extended by new functionality. The NEMEA framework is designed to allow quick and easy implementation of new modules. Although NEMEA can be used in production environment for analysis of live traffic, it is also designed to serve as a common platform for researchers in the area of network security and monitoring. It allows for fast prototyping of new traffic analysis methods, testing them on both offline and online data and comparing them with existing methods. Therefore, although we already provide a number of modules for common tasks and several detectors of malicious traffic, we hope a community will develop much more in the future.

The framework used by the NEMEA system is developed in C language and brings support for implementation of NEMEA modules in C, C++ or Python (with possibility to add more languages later). The system should run on any UNIX-like operating system.

B. Architecture

Figure 3 shows a simplified architecture of the NEMEA system. There is a set of running modules (interconnected by interfaces, which is not shown here). The set of modules can be controlled and monitored by a tool called Supervisor. All

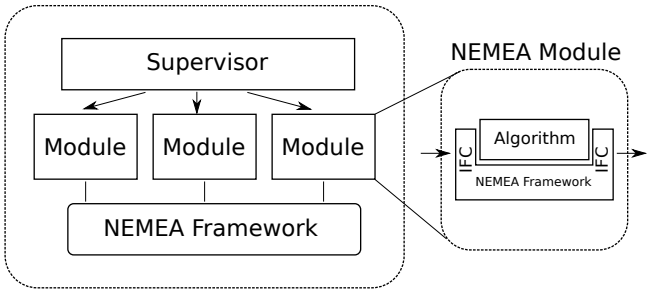

Fig. 3. High level look at the NEMEA system.

the modules are programs built upon NEMEA framework, *i. e.* the modules use functionality that is implemented in shared libraries of the NEMEA framework.

The right side of Fig. 3 points out usual inner organization of a module. Every module implements an algorithm or method for traffic analysis or detection and it uses features of the NEMEA framework for communication with other modules and for access to information contained in data records.

The most important part of the framework is *TRAP library* (Traffic Analysis Platform), which implements the communication interfaces and other basic functions needed by every NEMEA module. Another library – *UniRec* – implements a data format for binary representation of flow records and other information. UniRec is the default data format used for communication of modules. There is a Python wrapper around these two libraries that supplies API for modules written in Python. The last part of the NEMEA framework is the *common* library that provides a number of functions and data structures commonly used in network data analysis algorithms, such as various hash functions, hash tables, Bloom filter, prefix tree, or B+ tree.

The NEMEA system follows stream-wise concept of data processing. That means it is designed to process data continuously at real-time or near real-time without a need of a data storage. Writing and reading flow data to/from hard-drive is often a performance bottleneck in other systems for flow data analysis. In NEMEA, all the data remains in operation memory (unless some module intentionally stores them to disk or database), allowing real-time processing of data even from large networks using a single server. However, if needed, NEMEA can be distributed to more servers since each module can run separately and communicate with others via network.

C. Communication interfaces

TRAP Communication Interfaces (IFC) allow modules to communicate with each other. The IFCs are unidirectional, each one represents either input or output of the module. Each module can use multiple input and output IFCs. An output IFC of a module (*sender*) can be connected to one or more input IFCs of other modules (*receivers*). All receivers connected to the same sender get the same data.

The data are sent over an IFC as a potentially infinite stream of short messages (max 64 kB each). A message may be a flow record, result of a detection algorithm (alert), statistics computed from flow records in some time window or anything else. Formats of the messages are described later.

The IFCs are in fact an abstraction of several different underlying methods of interprocess communication. The two main ones are based on UNIX domain sockets and TCP sockets. The former one is used for communication between modules on the same system, the latter one for communication over a network. There are also two special types of IFCs – *file* and *blackhole*. The *file* type allows to store a stream of data into a file (when used as output IFC) and replay it later (when used as input IFC). The *blackhole* type can be used as output IFC only. It simply discards all messages sent to it.

The key point is that a data processing algorithm inside a module is completely abstracted from the type and parameters of module's IFCs. It just calls functions to receive data from or send to a specified interface. The types of interfaces to use, together with parameters specifying where they should be connected (socket name, IP address and port, file name), are passed as command-line parameters when the module is started. The parameters are processed by the TRAP library so the developer of the module do not need to care about it. The library also handles most of the IFC related errors that may occur, for example if a connection to the other module breaks (*e. g.* because the other module is restarted) the library automatically tries to reconnect.

Generally, NEMEA is designed in such a way that the developer of a module can focus on data processing algorithm only, leaving all the integration work up to the TRAP library. It significantly shorten the time needed to develop, test and deploy a new traffic analysis method. It also opens the system to less experienced programmers, *e. g.* researchers focusing rather on traffic analysis methods than on programming.

D. Data formats

Data are exchanged over IFCs in one of three supported formats – unstructured data, JSON and NEMEA's own binary format UniRec. The first two are rarely used, flow data as well as most of other data records are transferred in UniRec format.

UniRec is an efficient binary format for storage and transfer of simple data records similar to plain C struct. In addition to the C struct it supports fields with variable length. UniRec itself is a generic data structure, a particular format is given by *template*, *i. e.* a set of fields in a record.

In comparison to other formats for transfer of simple records like IPFIX, JSON or its binary equivalents BSON or MessagePack, UniRec has two key differences. First, it is designed to allow very fast access to fields of a record. While other formats have to be parsed before fields can be accessed, UniRec fields can be read directly, with access speed almost equal to plain C struct[2]. But contrary to C struct, an UniRec template can be defined at run-time. Second, all records sent over a single IFC have the same template. This is not a problem in most use cases (when it is, JSON can be used instead) and it significantly simplifies data processing.

The data format compatibility is checked automatically by IFCs. Each output IFC specifies the data format it is able to send, while each input IFC specifies a set of required fields. These formats can be specified at run time, which

[2]There is one additional memory access to a small table which easily fits into L1 cache.

adds to flexibility of the system. When two IFCs are about to be connected, their formats are checked. If the output IFC contains all the fields required by the input IFC, the connection is established and data transfer begins. Otherwise the connection is refused.

Therefore, if a module for processing HTTP traffic needs flow data with URL field and a user tries to connect it to an IFC providing only basic flows with no L7 data, the connection fails.

This mechanism is useful not only for error checking. For example, it allows to create a generic logging module which automatically recognize what data it receives. Using this information, the module knows how to interpret messages and how to log them.

We want to explicitly point out that NEMEA natively supports flow records extended by L7 information. UniRec is a generic format whose records can contain any fields, therefore, flow records can be naturally extended by any new information elements, even at run-time. In general, the system allows to add or remove modules at run-time without an interruption of other modules. This property is important for production deployment of complex configurations of the NEMEA system.

E. Central Configuration and Monitoring

Modules of the NEMEA system can be run manually as any other set of UNIX processes. However, the system can also be controlled and monitored centrally by a tool called *nemea-supervisor*, which is usually a better option, especially for instances composed of a large number of modules. Nemea-supervisor can run as a system daemon or in an interactive mode. It also supports configuration via the standard NETCONF protocol [13].

Nemea-supervisor takes care of modules according to an XML configuration file. The file defines modules, their parameters and a grouping to profiles – groups of modules that can be started or stopped together, *e. g.* for an experiment. The configuration can be changed at run-time using provided thin client or any NETCONF client.

As a monitoring tool, nemea-supervisor periodically retrieves state information of every module and, with respect to the configuration, performs actions needed to keep the modules running or stopped. Besides the module's status, nemea-supervisor reads some statistics about the resource consumption of modules (CPU and memory usage) and also about their interfaces. Every module's IFC automatically updates counters of received or sent messages and the counters are read by nemea-supervisor via a special *service IFC* opened in every module. All statistics about modules can be periodically exported into the Munin system [14].

F. System Performance

Overall performance of the NEMEA system depends mainly on a set of deployed modules and their resource requirements. However, there is a limitation given by maximal throughput of IFCs. Every output IFC uses a buffer to optimize utilization of data transfer in order to increase throughput.

Maximal number of messages per second (MPS) that can be sent depends on a message size. Fig. 4 shows a relation

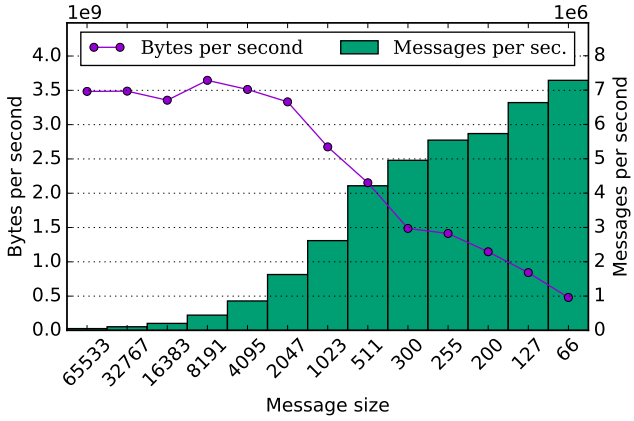

Fig. 4. Impact of message size on throughput of the socket based IFC.

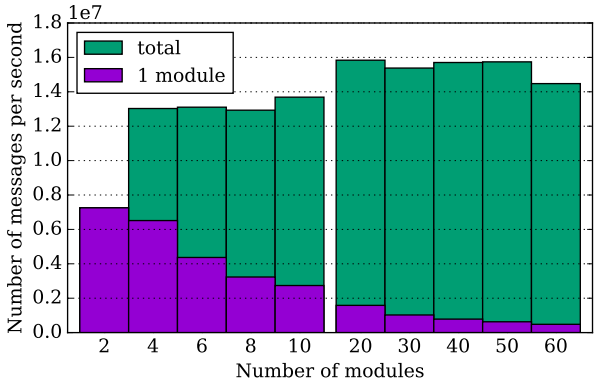

Fig. 5. Impact of number of running modules to IFC throughput. The number of modules (x-axis) contains all running modules (senders and receivers).

between MPS and message size measured between a pair of test modules (one sender and one receiver) on the same machine, whereas, messages were generated in memory. The green bars in the figure show the number of MPS that can be sent/received via IFC. The highest number of MPS (over 7 million per second) was measured for 66 B message size, which is the size of a basic flow record that we use. Due to significant overhead for this size of message, the transmission speed in bytes per second (purple line in the figure) is the lowest (over 480 MB/s). L7 information consumes additional memory besides the basic flow information. For the biggest measured messages (65 KB messages), the number of MPS is low (53 thousand MPS) but the transmission speed is about 3.5 GB/s.

Philosophy of NEMEA is based on running several modules at once. However, the more running modules, the less resources are available for each module. Figure 5 shows throughput of IFCs when running multiple pairs of modules. All the modules were sending messages of a fixed size (66 B was chosen). The purple bars represent the average number of MPS that was received by a single receiver, the green bars represent the sum of MPS from all receivers. For two modules (one pair), the total number equals to the MPS of one receiver.

The real-world performance of a complete system heavily depends on a particular set of instantiated modules and

complexity of algorithms they implement. However, as shown in Sec. V, it is not a problem to process flow data from a medium-sized NREN by tens of modules using only a single server.

IV. Targeted Real Use-Cases

NEMEA already contains a number of modules for different purposes, mostly for detection of various kinds of malicious traffic. This section describes main topics that are covered by the modules. Most of them are currently in use in CESNET2 network.

A. Detection up to L4

Network scanning belongs to the most common activities that occur in the Internet. It is usually harmless, however, it can be used by attackers to gather information. There are NEMEA modules for scans detection (port scans were analyzed in [15]).

Denial of Service (DoS) and Distributed DoS (DDoS) belongs to the most powerful types of attack. According to various reports (*e. g.* [16]), number of DoS/DDoS attacks increases and volume of traffic that attackers can generate gets higher. Detection and mitigation of these attacks is a challenging task, since it is very difficult to reliably and timely recognize malicious and benign traffic. The NEMEA system tries to get into the topic by providing several detection modules based on different analysis methods. For example, we implemented a detection method based on MULTOPS tree [17]. There is also a specialized NEMEA module for detection of amplification attacks.

B. Detection using L7

Session Initiation Protocol (SIP) can be used as a signalling protocol for Voice over IP (VoIP) that is a modern successor of telephone services. In specific circumstances, a misconfigured SIP Private Branch Exchange (PBX) allows to call to a PSTN phone number just by using the correct prefix, added to the number. Attacks trying to guess the prefix and make calls to premium-rate numbers are quite common and if successful, they can lead to a significant financial loss for the owner of the vulnerable PBX. A NEMEA module detecting this kind of attack using flow records extended by selected SIP headers was proposed in [18].

DNS protocol is usually not restricted by security policies and firewall settings because it belongs to indispensable services. It can sometimes be used to circumvent a network connection restrictions or escape from a secured network by encapsulating data into DNS messages and thus creating a communication tunnel. There is a NEMEA module for detection of such tunnels based on statistical analysis of DNS messages, further described in [19].

Heartbleed is a critical bug discovered in the OpenSSL library in 2014. It gives an opportunity for attackers to remotely read random chunks of memory from a server that uses the vulnerable version of OpenSSL. A NEMEA module for Heartbleed exploit detection was developed in a few days after the bug was published. It analyzes information from SSL protocol headers which are extracted by a special plugin for flow exporter. Within the first two months of operation, the module discovered more than a thousand vulnerable hosts in our network that were attacked (or probed) from the outside. The detection mechanism is described in detail in [20].

NEMEA can be used to detect devices infected with malware as well. The paper [21] presents that, having samples of malware, it is possible to retrieve valuable information for the detection of infected devices connected to the network. A generic *filter* module can be used to find a communication with suspicious servers. In such a scenario, extended flow records with domain names and URLs of HTTP are the input of the filter. The filtering condition can contain the suspicious addresses, domain names and URLs. Every matching connection is immediately reported, since its source is probably infected by the malware.

C. Alert Handling

The detection modules generate records about detected security incidents (alerts). To abstract the detectors from tasks related to alert handling, such as logging or reporting, NEMEA provides a means to handle the alerts in a unified way. This is represented by a set of modules called *reporters* that convert alerts from detectors into a unified format and then they can: log alerts into files, store alerts into database, send e-mails containing information from alerts, send alerts into the Warden[3] system.

Since these modules are implemented as Python scripts, it is easy to extend them to support any output data format or to export alerts to any other system.

D. Offline testing

NEMEA does not have to be used just in a production deployment processing live data. It can work offline and process stored data as well. There are modules for reading flow data from files in nfdump format, fastbit database used by IPFIXcol or CSV files. It is also possible to store and replay a stream of data generated by any NEMEA module directly in UniRec format.

This allows to repeatedly send the same data into a set of processing modules, which is useful for testing modules (and therefore processing methods) as well as for research, *e. g.* for comparison of different methods or parameter settings using the same input data.

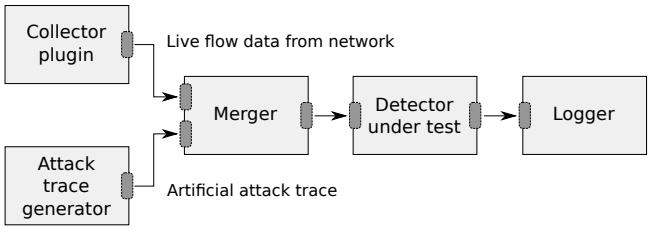

Fig. 6. Testing a detector by injecting artificial attack traces into a live stream of flow data

An interesting use-case is also a possibility to mix two or more streams of data using the *merger* module. For example a

[3]Warden is a system for sharing information about detected incidents within a community of security teams, developed at CESNET, https://warden.cesnet.cz/

stream of flow records got from normal traffic can be merged with a stream generated by an attack simulator or with a trace of an attack stored previously. Thus, attack traces can be mixed into normal traffic to test abilities of detection modules, as shown in Fig. 6.

V. Real world deployment and results

An instance of NEMEA system is deployed at CESNET since early 2014. It receives and analyzes flow data from probes deployed on the perimeter of CESNET2 network, that is ten lines with capacity ranging from 10 Gbps to 100 Gbps connecting CESNET2 to other backbone networks. The data are not sampled in any way. On average, the probes generate 120,000 flow records per second during peak hours.

At the time of writing (June 2016), our instance of the NEMEA system consists of 28 modules. All the modules run on a single server with 6 CPU cores and 12 GB of memory and are able to process all incoming data without any loss. In fact, no more than 40 % of server's resources (both CPU and memory) are utilized in normal circumstances.

The system is able to detect port scans, DDoS attacks (simple SYN floods as well as DNS and NTP amplification attacks), dictionary attacks on SSH, and watches for connections to several malicious IP addresses and URLs. Also, various statistics about the traffic are computed and stored. At last, one of the modules performs on-the-fly anonymization of the flow data and re-sends them to a server with less restricted access. This is used for development and testing of new modules by network security students and other academic people who can not get access to real data due to privacy concerns.

On average, every day the NEMEA system detects and reports the following events in the CESNET2 network:

- 110,000 horizontal port scans[4] (76 every minute).
- 12,000 dictionary or bruteforce attempts to log in to SSH (8.3 per minute).
- 2,400 DDoS attacks (mostly DNS and NTP amplification).

The number of DDoS attacks may seem very high. This is partly caused by the fact that a single attack may be reported by two modules and that long attacks may be reported several times due to properties of methods used for detection. Therefore, it is rather a number of alerts generated by detectors than real number of attacks. Nevertheless, even if the alerts are aggregated, the number of attacks is still in the order of hundreds. This is because most of the DDoS attacks today use thousands of DNS or NTP servers across the world for reflection and amplification of the attack traffic. If traffic to just one of the servers passes through the CESNET2 network, the attack can be detected. In fact, we often observe use of a single server for several separate attacks at the same time.

We also have several modules for analysis of L7 data present in our extended flow records, for example a detector of attacks on VoIP servers or detector of DNS tunnels. They are still considered experimental and are not running on the main NEMEA instance, but for example, the VoIP detector reports around 1,100 attacks per day if it gets data from all the probes.

We also often use L7 data for ad-hoc filtering by URL in HTTP or hostname in DNS requests based on current needs of security management. That means ad-hoc addition of modules for filtering and logging the traffic of interest. For example we recently acquired a sample of a new malware and analyzed it in cooperation with our forensic laboratory [22]. This resulted in a list of IP addresses and URLs used to control a botnet. A set of filter modules looking for those IP addresses and URLs in L7-extended flow data was immediately added to our NEMEA system. It revealed 11 infected devices in our network during 2 weeks. The detection was done continuously and alerts were sent at near real-time.

VI. Conclusion

Flow-based measurement and analysis became a standard approach for network monitoring. Traditional flow records provide visibility to transport layer (L4) only. However, for detection of some kinds of problems, application layer (L7) information is necessary. Although there exist exporters capable of parsing L7 information, it is hard to process them with current tools. We therefore created a new platform for analysis of L7-extended flow data – NEMEA.

The NEMEA serves for both experimental and operational use. One of its most important features is the capability of L7 processing. At the same time, it is a flexible modular system that allows researchers and network operators to extend its functionality by implementing new NEMEA modules. It can also be viewed as a common platform for development of traffic analysis algorithms, which allows to easily test them on both offline traces and live data, compare to other algorithms and eventually deploy them operationally. The modularity of NEMEA also makes the system scalable to handle even large numbers of flow records. In case one machine does not have enough resources, it is possible to distribute the computation, *i. e.* start NEMEA modules on multiple hosts.

In this paper, we have shown a complex deployment combining NEMEA system with a set of high performance exporters and an open-source collector IPFIXcol. However, since NEMEA contains its own minimalistic exporter, it can also be used independently on smaller networks.

Deployment at CESNET2 network have proven that NEMEA can be successfully used to monitor large networks and detect various kinds of malicious traffic. Thousands of incidents have been detected thanks to NEMEA.

Acknowledgment

This work was partially 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 Technology Agency of the Czech Republic under No. TA04010062 *Technology for processing and analysis of network data in big data concept*.

[4]To avoid false alerts, we set thresholds quite high. Port scan is reported when more than 200 connections on different destinations are attempted within 5 minutes.

REFERENCES

[1] R. Hofstede, P. Celeda, 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, 2014.

[2] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of IP flow-based intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 3, pp. 343–356, 2010.

[3] L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras, "SSHCure: A flow-based SSH intrusion detection system," in *Dependable Networks and Services*. Springer, 2012, pp. 86–97.

[4] Flowmon Networks, "The Most Powerful NetFlow Probes in the World." [Online]. Available: https://www.flowmon.com/en/products/flowmon/probe

[5] CESNET, "IPFIXcol." [Online]. Available: https://github.com/CESNET/ipfixcol/

[6] V. Bartoš, M. Žádník, and T. Čejka, "Nemea: Framework for stream-wise analysis of network traffic," CESNET, a.l.e., Tech. Rep., 2013. [Online]. Available: http://www.cesnet.cz/wp-content/uploads/2014/02/trapnemea.pdf

[7] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer networks*, vol. 31, no. 23, pp. 2435–2463, 1999.

[8] M. Roesch and *et. al.*, "Snort: Lightweight intrusion detection for networks," in *LISA*, vol. 99, 1999, pp. 229–238.

[9] P. Haag, "NFDUMP – Netflow processing tools." [Online]. Available: http://nfdump.sourceforge.net/

[10] ——, "NfSen – Netflow Sensor." [Online]. Available: http://nfsen.sourceforge.net/

[11] CERT/NetSA at Carnegie Mellon University, "Analysis Pipeline." [Online]. Available: {http://tools.netsa.cert.org/analysis-pipeline}

[12] ——, "SiLK (System for Internet-Level Knowledge)." [Online]. Available: http://tools.netsa.cert.org/silk

[13] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)," RFC 6241, Internet Engineering Task Force, Jun. 2011.

[14] "Munin." [Online]. Available: http://munin-monitoring.org

[15] T. Cejka and M. Svepes, *Analysis of Vertical Scans Discovered by Naive Detection*. Munich, Germany: Springer International Publishing, 2016, pp. 165–169. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-39814-3_19

[16] Kaspersly Lab, "Kaspersky DDoS Intelligence Report Q3 2015," November 2015. [Online]. Available: https://securelist.com/analysis/quarterly-malware-reports/72560/kaspersky-ddos-intelligence-report-q3-2015/

[17] T. M. Gil, "MULTOPS: A data structure for denial-of-service attack detection," Ph.D. dissertation, Vrije Universiteit, 2000.

[18] 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 (LNCS 9122)*. Springer International Publishing, 2015, pp. 87–99.

[19] T. Cejka, Z. Rosa, and H. Kubatova, "Stream-wise detection of surreptitious traffic over DNS," in *19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE, 2014, pp. 300–304.

[20] V. Bartoš, "Heartbleed Detection at CESNET using Extended Flow Monitoring," in *Proceedings of 8th International Scientific Conference on Security and Protection of Information*, 2015.

[21] T. Cejka, R. Bodó, and H. Kubatova, "Nemea: Searching for Botnet Footprints," in *The 3rd Prague Embedded Systems Workshop (PESW2015)*, 2015.

[22] CESNET, "FLAB – Forensic laboratory." [Online]. Available: https://flab.cesnet.cz/