FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF BACHELOR'S THESIS

Title:	Evaluation of captured flow data of suspicious devices
Student:	Jan Suchara
Supervisor:	Ing. Tomáš Čejka, Ph.D.
Study Programme:	Informatics
Study Branch:	Computer Security and Information technology
Department:	Department of Computer Systems
Validity:	Until the end of summer semester 2019/20

Instructions

Study the current technologies of network traffic monitoring based on flow data, especially the NEMEA system [1] and the new set of modules - the Adaptive Filter (AF) by Filip Suster [2].
Study the principle of the "Scenarios" functionality of AF and analyze its output data.
Design an algorithm for automatic processing and evaluation of the AF output data to compute statistical information (e.g., unique destination IPs, frequency of data flows, or services used by suspicious IPs) and provide a report for security teams.
Implement the algorithm as the Evaluator module for the AF.
Test the functionality, precision, and performance of the developed module.

References

[1] T.Cejka, et al.: "NEMEA: A Framework for Network Traffic Analysis," in 12th International Conference on Network and Service Management (CNSM 2016), Montreal, Canada, 2016
[2] Šuster, Filip: Automatická detekce podezřelého síťového provozu pomocí blacklistů. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019

prof. Ing. Pavel Tvrdík, CSc.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague January 28, 2019

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Evaluation of Captured Flow Data of Suspicious Devices

Jan Suchara

Department of Computer Systems
Supervisor: Ing. Tomáš Čejka, Ph.D.

May 16, 2019

Acknowledgements

I would like to express my gratitude to my supervisor, Ing. Tomáš Čejka, Ph.D., and to Ing. Filip Šuster for their guidence and help with this thesis. My thanks also go to my family and my girlfriend for their selfless support during my studies.

Declaration

In Prague on May 16, 2019 …………………

Citation of this thesis

Suchara, Jan. *Evaluation of Captured Flow Data of Suspicious Devices.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

Tato práce se zaměřuje na analýzu síťového provozu klientů komunikujících s adresami na veřejných blacklistech. Hlavním cílem bylo rozlišit atributy provozu, které mohou být použity k rozeznání bežného provozu od provozu nakažených zařízení. Výsledkem práce je modul Evaluator pro systém NEMEA, který rozšiřuje existujicí sadu modulů blacklistfilter. Evaluator počítá statistiky provozu podezřelých adres a využívá námi získané výsledky k omezení počtu false positive hlášení.

Klíčová slova analýza síťového provozu, detekce botnetů, C&C, IPFIX, NEMEA

Abstract

This thesis focuses on the analysis of traffic generated by clients communicating with addresses on public blacklists. The main goal was to identify traffic attributes which could be used to differentiate the malicious traffic from benign traffic. The result of this work is a module for the NEMEA system—Evaluator. The module extends the functionality of existing module set blacklistfilter. Evaluator is designed to determine statistics of suspicious traffic and uses results of our measurements to reduce the number of false positive alerts.

Keywords network traffic analysis, botnet detection, C&C, IPFIX, NEMEA

Contents

List of Figures

List of Tables

Introduction

With the Internet's rapid growth and its expanding user base, the number of attacks on connected devices rapidly rises. Attackers' motives are diverse: ideological, political or pure curiosity. However, the primary motivation remains the same—financial interests. Despite the extensive campaign, raising public awareness, and countermeasures, new threats still emerge. Practically every device connected to the internet poses a threat.

In recent days, we have been observing an increase in attacks targeted on smartphones or IoT devices. These gadgets often works out-of-the-box, and users neglect their proper configuration. Nevertheless, users are not always who is to be blamed. Nowadays, it is common that hacker groups or secret services offer a substantial amount of money for discovering zero-day vulnerabilities. As an effect, malware becomes increasingly sophisticated, and its detection more challenging. Attackers take advantage of new technologies and approaches, some of which were created to protect regular users, e.g., encryption, faster hardware for password cracking.

One of the most widespread and potentially destructive threats are botnets. These networks of malware-infected devices—*bots*—are remotely controlled by hackers—*botmasters*. Botmasters use bots in various ways like launching distributed denial-of-service attacks, stealing personal information, or spreading other malicious applications. Hence, it is desirable to discover these malicious hosts as soon as possible so they could not do more harm to other parts of the infrastructure. Detection of bots is an immensely complex topic researched by experts all over the world. Several approaches, like signature-based or anomaly detection, are used. These techniques often combine DPI (deep packet inspection), machine learning algorithms and network flows analysis. The flow analysis could be a reliable tool for early detection of possible threats. It could be used in situations where DPI fails due to encryption, its demands on computation power, or when the privacy of users has to be preserved. Several platforms for analysing network flows exist.

In this thesis, we set to analyse the traffic of clients reported as suspicious by existing NEMEA modules. Our main task was to develop a new module which would measure statistics about said hosts and then generate a report for a response team. For this purpose, we compared attributes of malicious botnet communication with those of regular traffic. During our experiments, we identified which characteristics differentiate infected hosts from benign ones. Furthermore, we developed a method that could help to decrease the number of false positive alerts. The results led to the creation of Evaluator module, which was integrated with the existing solution.

Motivation

The main goal of this thesis is to analyse suspicious clients' traffic and summarization of its attributes. Thanks to the blacklistfilter module set, the result of Ing. Filip Šuster's work, we could collect flows of devices communicating with entities on publicly available blacklists. The module generated a basic report about each incident. We set to extend its functionality to provide more information. As a result, a new module Evaluator was created. It is used to enrich blacklistfilter's alerts with additional information about the monitored clients. Consequently, it is easier to decide whether the traffic is regular or if the device represents a real threat.

The first chapter describes the concept of network flows, principles of its collection. Section 1.2 and Section 1.5 are dedicated to the NEMEA system, its surrounding infrastructure, and details about the blacklistfilter modules. Approaches to suspicious clients' traffic analysis and its results are presented in Chapter 2. The final part of the thesis describes the design and the implementation of new modules for traffic analysis.

Chapter 1

Analysis

This chapter introduces the concept of network flows and describes principles of their collection. The following section presents the NEMEA System and its essential components designated for building flow-analysis programs. Finally, I explain the work of Ing. Filip Šuster on whose results this thesis builds upon. Comprehension of mentioned topics was necessary for the fulfilment of defined goals.

1.1 IP Flow and Export Protocols

Flow traffic measurement is a widely used technique for monitoring large networks. It provides insight into information flows inside these networks and may help with bottleneck identification or nefarious activity detection.

A flow can be defined as a set of IP packets which share common attributes in Internet and Transport layer headers, typically source IP address, destination IP address, source and destination port number, and transport layer protocol. Other characteristics (such as packet count, bytes transferred, flow duration, and more) may also be collected [1].

Packets are assigned to flows when they pass a point in a network called *flow exporter*, usually a router or L3[1] switch. Whenever a flow expires (i.e., no additional packets of the given flow were received during the predefined time window) a flow record containing aggregated information is generated. This record is sent to a *flow collector* where it is stored for offline analysis. *Flow analysers* are then able to calculate various statistics from stored flows and present them to the operator.

Exporting protocols define flow collection mechanisms, which attributes are to be monitored, and details of communication between collectors and exporters. As of today, several of these protocols developed by various vendors

[1]OSI Layer 3 (Network Layer)

are in existence, most notably IPFIX (IP Flow Information Export) [2] and Cisco's NetFlow v9 [3].

1.2 NEMEA System

NEMEA, which stands for Network Measurements Analysis, is an open-source modular system for IPFIX flow processing developed in CESNET. It provides network administrators and security experts with a framework for automated real-time analysis of network traffic [4].

Building blocks of the system are the so-called *modules*—independent units running as separate processes. Each module has a predefined set of input and output Traffic Analysis Platform (TRAP) interfaces which are used to exchange data with other modules. Every interface is one-way only and transmits data in a custom format defined in Unified Record (UniRec) protocol or JSON in a stream-wise manner, i.e., one record at a time.

This design makes extending the system with more modules simple and provides users with the ability to assemble it correspondingly to their specific needs.

1.2.1 Traffic Analysis Platform

As previously mentioned, all modules inside NEMEA use TRAP interfaces for exchanging data. These interfaces are implemented inside the `libtrap` dynamic library. Whenever a module is loaded, the library handles initialization of interfaces based on parameters given to the process. TRAP interface can be one of the following types[2]:

- Unix domain socket
- TCP interface
- TLS interface
- Blackhole interface[3]
- File interface

Additionally, the library allows the user to set 3 more attributes of each interface, e.i., buffering strategy, timeout, and interval of buffer emptying.

[2]https://nemea.liberouter.org/trap-ifcspec/
[3]Equivalent of Unix null device

1.2.2 Unified Record

Unified Record (UniRec) is a binary data format used by TRAP interfaces. A *template* defines structure of each record and describes which *fields* are inside a single record. Type of fields denotes which information it can hold. One template is assigned to an interface at initialization and must not change during the interface's lifetime. Therefore, the format of exchanged data remains constant. The list of currently supported data types inside a UniRec record spans from simple integers and floats to MAC and IP addresses or strings of variable lengths.

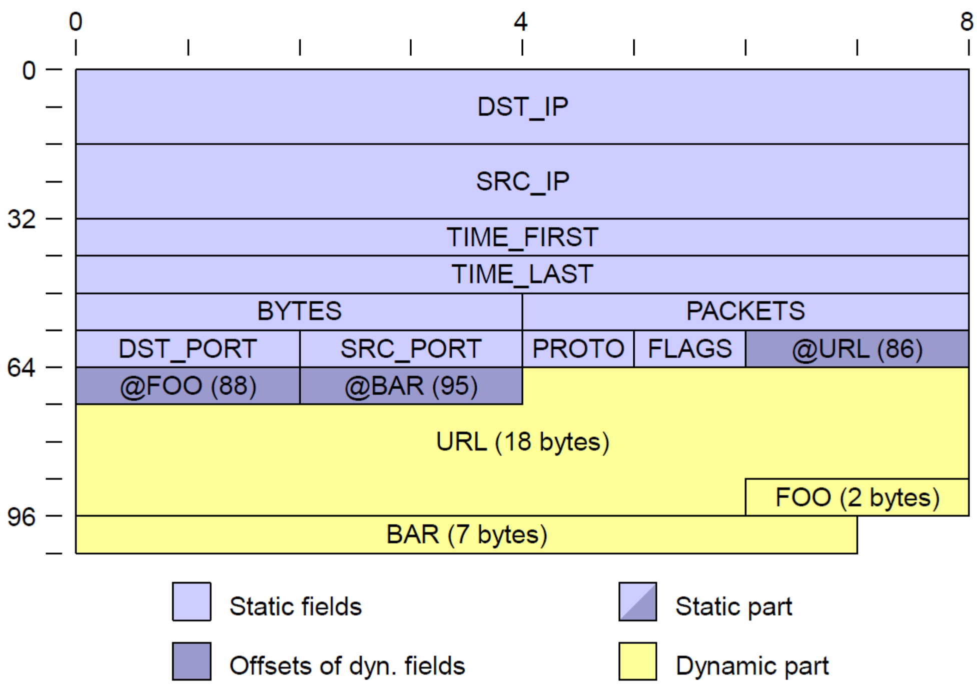

Figure 1.1: Illustration of UniRec record representing an IP flow with custom fields [5]

1.3 Warden

Warden[4] is a system based on a server-client architecture developed at CESNET. It is used for sharing information about network anomalies and threats between security teams. The central server receives events from *sending clients* (e.g., alerts from intrusion detection system, honeypots, or NEMEA modules) and ensures their redistribution to *receiving clients*. Messages exchanged be-

[4]https://warden.cesnet.cz/en/architecture

tween the server and clients are in a custom format—IDEA (Intrusion Detection Extensible Alert). The system is designed currently deployed in the CESNET2 network.

1.4 NERD

Network Entity Reputation Database (NERD) is a modular system serving as a source of information about IP addresses, networks, domains and other network entities. It is designed to collect alerts about malicious activities from various detection systems like Warden. It provides additional information, e.g., geolocation, whether the address acts as a TOR exit node. Furthermore, the database contains all detected incidents related to each entity. A *reputation score* is determined for each record. The score expresses the ”probability of future attacks combined with their expected severity” [6]. An Entity is removed from the system whenever no malicious activity was detected for a given period of time. A REST API is provided to query the database for information about individual entities in the JSON format.

1.5 Blacklistfilter Modules

This thesis sets out to examine and further analyse the output of an existing set of NEMEA modules named *blacklistfilter*. Its purpose is to detect possibly malicious communication between clients and blacklisted entities outside of our network, capture all of the related traffic, and report the event. Ing. Filip Šuster created Blacklistfilter as a result of his master thesis. The following section is based on his work [7] and explains the basic principles of individual modules.

1.5.1 Blacklist Downloader

Downloader is a Python module that periodically gathers blacklists from sources defined in its configuration file. Blacklists are of the following types:

- IPv4
- IPv6
- URL
- DNS

Plain text or CSV files are supported. Following data download, the module merges individual lists, removes duplicities, and sorts the records. The last step is crucial for the correct functioning of other modules. Subsequently,

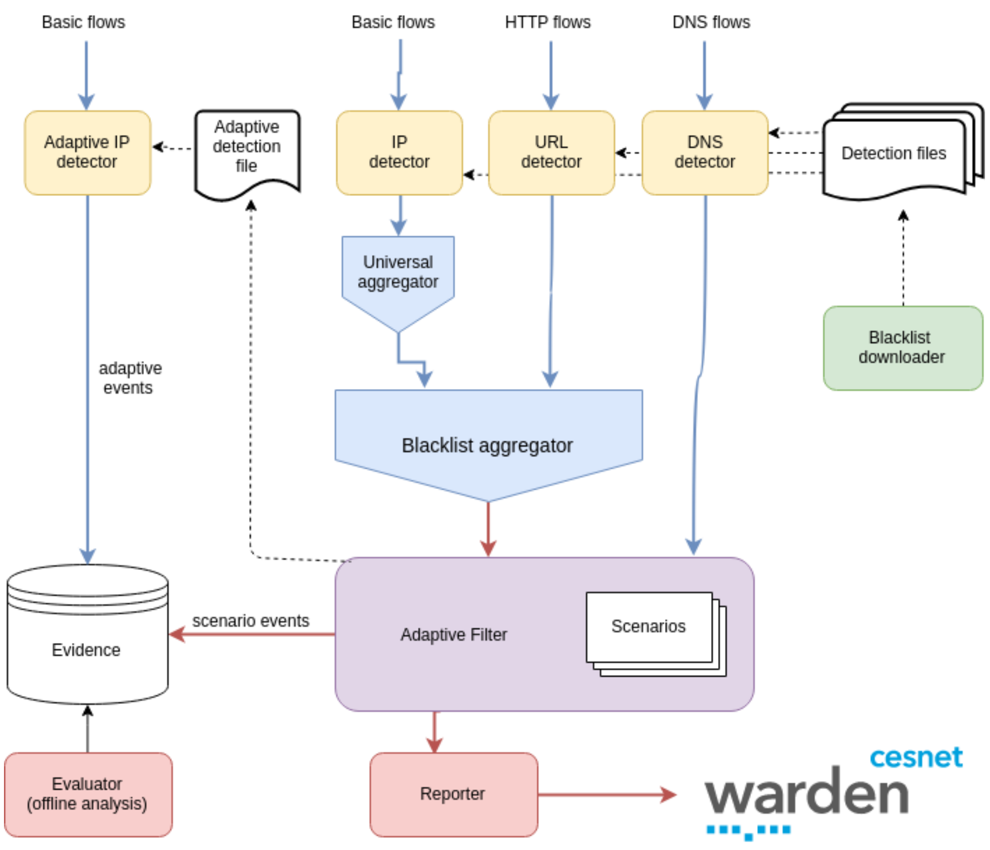

Figure 1.2: Blacklistfilter modules structure [7]

blacklists are written to files according to their type, e.g., ip4.blist, url.blist, etc.

Another useful feature of the downloader is versioning of used blacklists. Hence, it is possible to track how they changed in time and determine in which version an entity was added or removed.

1.5.2 Detector Modules

These modules are the core of blacklistfilter. They are responsible for the detection of malicious flows. The emphasis was on performance and effectiveness; therefore they were written in C/C++. Each detector has an input interface on which it listens for incoming flows in UniRec format. Received flows are tested whether they originate from a blacklisted entity. Whenever a match is found, various fields (depending on the type of the detector) are added to the record. Afterwards, it is sent to the output interface. There are 3 types of detectors:

- IP detector

- URL detector
- DNS detector

Detectors have a dedicated thread which listens for changes in blacklist files. Every time a file is edited, e.g., downloader updates it, modules load it into their memory.

IP detector inspects destination and source address of each flow. It uses binary search to traverse IP blacklist file. Records sent to the output interface are extended with `SRC_BLACKLIST` and `DST_BLACKLIST` fields. 64-bit numbers inside these fields represent bitmaps. They indicate on which blacklists was the source or the destination address found.

URL and DNS detectors monitor clients' requests for blacklisted domains. Their output UniRec template includes fields with additional HTTP and DNS protocol data.

Adaptive IP Detector

The function of this module is almost identical to the IP Detector. It collects all flows concerning clients previously communicating with malicious hosts. The adaptive detector has its own blacklist file which is managed by Adaptive Filter. IPs of suspicious clients inside our network are added or removed dynamically. Monitored IP is removed when a specific time from the last flow to a blacklisted entity has elapsed; hence, no more flows are captured. The output of this module is analysed by Evaluator and will be described in a later chapter.

1.5.3 Adaptive Filter

Adaptive Filter is a Python module which collects data from IP/URL aggregators and DNS detector. Each incoming record is called a *detection event*. Upon receiving a detection event, dedicated thread inside this process checks whether any prior communication with a particular *key* (blacklisted entity) was observed. If no match is found, a new instance of *scenario* event is created, and module signals Adaptive IP Detector to start capturing flows of the client. Otherwise, the corresponding instance is updated. A scenario is an essential aspect of the design. It is a class which holds information about individual detections and determines which action will be performed. The Adaptive Filter module can be easily extended with new scenarios and existing ones can be modified or removed, depending on the current need.

Adaptive Filter periodically measures elapsed time since the first detection event for every scenario. If it exceeds *evidence timeout* (specified among the module's parameters), a report in the JSON format is sent to the output interface and clients associated with this event ends are no longer monitored. The

```
{
   "first_detection_ts":1554494682.327665,
   "key":"1.2.3.4",
   "id":"a4e60be0-5669-414a-a84d-75804cad79fe",
   "event_type":"BotnetDetection",
   "grouped_events":[
      {
         "ts_first":1312967397.858,
         "source":"173.192.170.88",
         "ts_last":1312984101.404,
         "src_sent_bytes":324964,
         "type":"ip",
         "targets":[
            "111.222.111.222"
         ],
         "src_sent_packets":1493,
         "tgt_sent_flows":359,
         "source_ports":[
            80
         ],
         "protocol":6,
         "agg_win_minutes":0.1,
         "blacklist_id":2,
         "src_sent_flows":303,
         "tgt_sent_bytes":343824,
         "tgt_sent_packets":4108
      }
   ],
   "last_detection_ts":1554494682.327667,
   "grouped_events_cnt":1
}
```

Figure 1.3: Example of a scenario event in JSON

report holds information about the blacklisted address, clients communicating with it, and statistics about the traffic aggregated during the aggregation window.

Chapter 2

Dataset analysis

The main goal of this thesis is to analyse the traffic of suspicious clients, i.e., those communicating with blacklisted addresses, and generate a report for a response team. The report should summarise statistics which would help to decide whether the threat is real or not. We developed this thought further. If we could identify characteristics differentiating benign traffic from malicious, we could report only clients whose communication mimics that originating from known infected hosts. Consequently, the number of false positive detections could decrease. A host could be incorrectly classified as malicious when connecting to a legitimate service running on a blacklisted address or after initiating a horizontal scan of network with that address.

For this purpose, it was necessary to analyse the traffic of botnet samples. We used publicly available dataset CTU-13. The next step was to compare its characteristics with those of average traffic inside a network where would be our tools deployed. This demanded the creation of another dataset which represents communication inside the CESNET2 network. Both sets of data are described in the following sections.

CESNET2 dataset

The CESNET2 dataset was created from the real network traffic at the perimeter of the Czech national research and education network infrastructure, i.e., the dataset contains all traffic from CESNET2 to foreign networks, and vice versa, and transit traffic that went via CESNET2 bordering links.

There are 8 bordering links at the perimeter of CESNET2. Each is being monitored by a monitoring probe—flow exporter—that aggregates packet data into IPFIX records. The IPFIX data were collected on a collector server and converted into UniRec format to be easily processable using NEMEA system [4] that is deployed on the collector for traffic analysis and anomaly detection.

The dataset was anonymized using a Crypto-PAn [8] method implemented in the *anonymizer* module of the NEMEA system. Generally, this method transforms all IP addresses (with respect to the protocol version, i.e., IPv4 and IPv6) so they are no longer associated with real network prefixes and hosts. However, the method preserves relations between original addresses that communicated between each other.

Network traffic contained in the dataset was captured in 7 hours from all monitoring probes. The traffic consisted of 1.2 billion flow records that transferred 64.6 billion packets and 65.8 TB. Besides traditional NetFlow fields (`SRC_IP`, `DST_IP`, `SRC_PORT`, `DST_PORT`, `PROTOCOL`, `TCP_FLAGS`, `TIME_FIRST`, `TIME_LAST`, `BYTES`, `PACKETS`), there are additional fields: `TTL`, `TOS`, `LINK_ID`, `DIR_ID`.

CTU-13 dataset

Malicious traffic was obtained from CTU-13 Dataset created by a team at Czech Technical University for Stratosphere IPS project. The dataset contains communication of 13 botnet samples. Each scenario consists of the traffic originating from the malware, other hosts in the network and background traffic. Scenarios are of various length and show malicious activities of one or multiple infected machines (C&C communication, port scanning or spam distribution). The experiment was carried in a laboratory environment. Hosts ran as virtual machines running Microsoft Windows XP SP2 and were bridged into to university network. The traffic was being captured on the VM's hosts and on the router to include the flows of the background traffic [9]. The authors provided PCAP[5] files with separated botnet related communication, which was ideal for our purposes.

2.1 Approach

The goal of our experiments was to find out whether the attributes of benign and malicious communication differ and how. Before we could compare the two datasets mentioned earlier, it was necessary to filter data in CESNET2 dataset. At first, all the transient traffic contained in it was removed. Afterwards, we had to identify potentially infected clients (i.e., those communicating with blacklisted addresses) and separate them from the rest of the hosts. For this purpose, we created an aggregated blacklist from those downloaded by blacklist downloader in the course of data collection. We used the following publicly available IP blacklists:

- ZeuS Tracker[6]—a blacklist of C&C servers of Zeus crimeware kit

[5]Packet Capture

[6]https://zeustracker.abuse.ch/

Table 2.1: Distribution of traffic for each scenario in the CTU-13 dataset [9]

ID	Background	Botnet	Normal	Bots
1	10,124,854 (95.40%)	94,972 (0.89%)	392,433 (3.69%)	1
2	6,071,419 (95.59%)	54,433 (0.85%)	225,336 (3.54%)	1
1	4,381,899 (94.60%)	75,891 (0.49%)	744,270 (4.89%)	1
4	3,895,469 (91.91%)	6466 (0.15%)	336,103 (7.93%)	1
5	416,267 (91.37%)	2129 (0.46%)	37,144 (8.15%)	1
6	2,031,967 (94.12%)	4927 (0.22%)	121,854 (5.64%)	1
7	425,611 (93.71%)	293 (0.06%)	28,270 (6.22%)	1
8	11,451,205 (95.47%)	12,063 (0.10%)	530,666 (4.42%)	1
9	6,881,228 (90.22%)	383,215 (5.02%)	362,594 (4.75%)	10
10	4,535,493 (87.54%)	323,441 (6.24%)	321,917 (6.21%)	10
11	119,933 (29.33%)	277,892 (67.97%)	11,010 (2.69%)	3
12	119,933 (29.33%)	277,892 (67.97%)	11,010 (2.69%)	3
13	1,218,140 (93.76%)	21,760 (1.67%)	59,190 (4.55%)	1

- Feodo Tracker[7]—blacklist of C&Cs associated with Feodo malware family, i.e., Dridex, Emotet/Heodo
- Ransomware Tracker[8]—blacklist of IPs of ransomware sites and botnet C&C servers
- Malc0de[9]—a daily updated list of domains that have been identified as distributors of malware

Table 2.2: Number of IP addresses on blacklist during the time interval of the dataset

Blacklist	Number of IPs
ZeuS Tracker	113
Feodo Tracker	318
Malc0de	80
Ransomware Tracker	347
Total	858 (856 unique)

[7]https://feodotracker.abuse.ch/blocklist/
[8]https://ransomwaretracker.abuse.ch/
[9]http://malc0de.com/bl/

The resulting list was passed to Adaptive IP Detector as it was re-run again over the captured data. As a result, we obtained all the clients whose traffic could be considered suspicious. The approach identified 628 IPs communicating with 67 blacklisted addresses transferring 33,555 flows (83,096,746 bytes) in total. That represented only 0.003% of the overall traffic. In this phase, we were left with about 803,000 addresses. This number is significantly higher than the actual count of active hosts inside the network. We observed more than 648,000 clients with no outgoing and an insignificant number of incoming flows. The probable cause was a horizontal scan of the CESNET2 network—the flow-collector records flows whether the destination device responses or not. These addresses were also removed. The remaining data representing circa 159,000 clients were used to calculate average per-IP statistics of the CESNET network and compared with those of the CTU-13 dataset.

For the extraction of information, we utilised blacklistfilter, existing tools provided by the NEMEA system and the Evaluator module described in Chapter 3. A list with all IPs from the CENSET2 range was generated and passed to Adaptive IP Detector which processed captured data again. The module's output consisted of individual clients' records marked with a unique ID defined in the said list. Lastly, the Evaluator was used to analyse the result. The extraction of information from the malware dataset required conversion of PCAP files to the UniRec format. This was achieved by utilising the program `flow_meter` provided by the NEMEA Framework. Afterwards, we repeated the same steps.

We were interested in attributes that are not dependent on the length of time the traffic capture lasted, e.g., number of transmitted bytes, packets or flows. Reason for this was that the duration of available malware traffic samples varied significantly. Observed average characteristics were measured for both incoming and outgoing communication:

- Bytes per packet
- Packets per flow
- Packets per second
- Bytes per second
- Ratio of sent and received flows, packets, and bytes
- Count of unique IPs and ports contacted

Average of transmitted packets and bytes per second were estimated from the overall duration of observed flows divided by the sum of these attributes.

2.2 Results

The results of measurements described in the previous section were analysed in statistical software R Studio [10]. Data representing datasets were loaded into the program as two data frames and means of all observed attributes were calculated. Despite the knowledge of a decreased robustness for the data with unknown statistical distribution, we conducted statistical hypothesis tests using the Welch's t-test to determine whether we could distinguish the legitimate traffic from botnet traffic based on data available. Both used datasets contain a sufficient number of observations which means that yielded results should be asymptotically approaching the real values. More rigorously, we used the two-sample Welch's t-test to determine whether means of the value extracted from the benign traffic μ_{legit} were statistically significantly different from the those of botnet traffic μ_{bot}, i.e.,

$$H_0 : \mu_{\text{bot}} = \mu_{\text{legit}},$$
$$H_1 : \mu_{\text{bot}} \neq \mu_{\text{legit}},$$

Table 2.3: Results of statistical tests

Attribute	μ_{bot}	μ_{legit}	p-value
Bytes received/sent	1.5640	78.1806	<0.0001
Packets received/sent	0.2209	47.4944	<0.0001
Flows received/sent	0.3421	57.5497	<0.0001
Packets per flow received	12.6743	46.8425	<0.0001
Packets per flow sent	52.4224	61.1361	0.8077
Packets per second received	2.8825	78.4948	<0.0001
Packets per second sent	5.8640	68.8428	<0.0001
Bytes per packet received	645.9626	431.2462	0.0546
Bytes per packet sent	576.9199	143.7808	<0.0001
Bytes per second received	3235.4860	15965.369	<0.0001
Bytes per second sent	3699.904	6657.938	0.2175
Unique IPs contacted	1964.1429	159.8249	0.0336
Unique ports contacted	22689.8857	161.9317	0.0002

The said test is implemented as an R function `t.test`. Means μ_{bot}, μ_{legit} and results are presented in Table 2.3. We can see that the p-value is lower than 0.05 for several measured attributes. Hence, we rejected the hypothesis H_0 at the level of significance 0.05. These attributes are:

- ratios of sent and received bytes, packets and flows
- number of packets per sent flow
- bytes per packet sent
- number of unique IPs and ports contacted

These results led us to the conclusion that these characteristics could be used to identify malicious hosts inside out network. The question is whether these attributes remain the same for other types of malware or how would they differ in other networks.

Based on our observations, we proposed a method which could be used to decrease the number of false positive alerts generated by Adaptive Filter. We utilised the results to calculate thresholds for each of the attributes distinguishing bots from benign hosts.

To find this threshold, we determined the median value of the attribute for both types of traffic. We chose a median for its statistical property ensuring that a half of observations lie above it and the second half is located below it. The median of one group was always smaller than the other but the method would work even if it was not. Then, we iterated over the interval between them in steps of size 0.01. In each step, we determined the percentage x of observations which exceeded the current value from the group with lower median. For the other group, we were interested in the percentage y of observations whose value was below the value. Afterwards, we calculated the value x/y. The threshold was the value fulfilling the criteria that x/y should be as close to 1 as possible. This approach minimises the number of incorrectly categorized hosts. Graphical representation of our approach is shown in Figure 2.1.

A client was marked as malicious whenever thresholds of 5 and more attributes were exceeded. The method was implemented in Evaluator module and its prediction accuracy was tested on both datasets. Results of the tests are presented in Chapter 5.

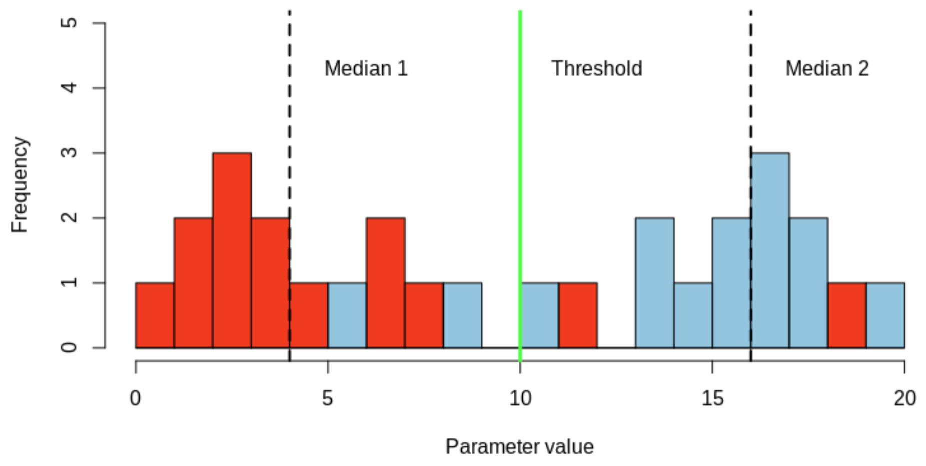

Figure 2.1: Visualisation of a threshold value lying between medians 1 and 2 of an attribute for given groups, i.e. botnet and benign traffic

Chapter 3

Design

In the following chapter, I will describe the design of the newly created NEMEA modules **Evaluator** and **Split evidence**. The modules were created as a result of this thesis. Evaluator serves as a replacement for a script in Adaptive Filter which provided only basic information about detection events. Its purpose is the processing of information received from Adaptive Filter and analysis of flows from suspicious clients. Split Evidence is a supportive module which converts flows into a form readable by Evaluator.

Changes in Evidence

Initially, it was necessary to propose a method of storing the output of Adaptive IP Detector and Adaptive Filter. The existing implementation kept most of the information inside the so-called *evidence*. The evidence consisted of two files:

- evidence_detection
- evidence_adaptive

The first file contained details about scenario events detected by Adaptive Filter. The latter one held captured flows of targeted clients.

This solution was not suitable for our purpose because we needed to store information related to individual events separately. Mentioned behaviour also made the analysis of data very time consuming because it was necessary to read the whole file each time. It also meant that the files could not be deleted while Adaptive Filter was in operation. This issue needed to be addressed, because the size of the evidence could increase rapidly.

A new module—*Split evidence*—was created to cope with this problem. Split evidence receives data from Adaptive IP Detector and sorts them into files dedicated for individual scenarios. With this module, the emphasis is on its efficiency. It has to process flows as fast as it receives them. Otherwise, it could cause a data loss or a decrease in the detector's performance. Hence, it was implemented in C/C++.

Evaluator

This module processes observed flows of suspicious clients, determines statistics about the traffic, and generates reports for the detection system. The important aspect of its design is that it decides which alerts received from Adaptive Filter are relevant and which are so-called false positives. The module has predefined thresholds for each of the identified characteristics differentiating normal traffic in the CESNET2 from the one generated by bots. Their values were estimated based on the approach described in Section 2.2. The following list summarises all information which the module collects about both incoming and outgoing flows and could be used to identify anomalies:

- Count of unique IP addresses and ports contacted by the client
- Number of received and sent flows
- Total size of transmitted data in bytes
- Number of packets and their average size
- Packets and flows per second
- Average flow duration

The monitored host is reported as malicious whenever 4 or more attributes of its traffic reach given thresholds. In the other case, the report is not sent into the detection system. Instead, it is stored together with the related data. This approach helps to decrease the number of false positive detections but preserves information needed in the case of other incidents. The Evaluator also search the NERD database for details about the blacklisted address. Finally, a detection alert in JSON format is created.

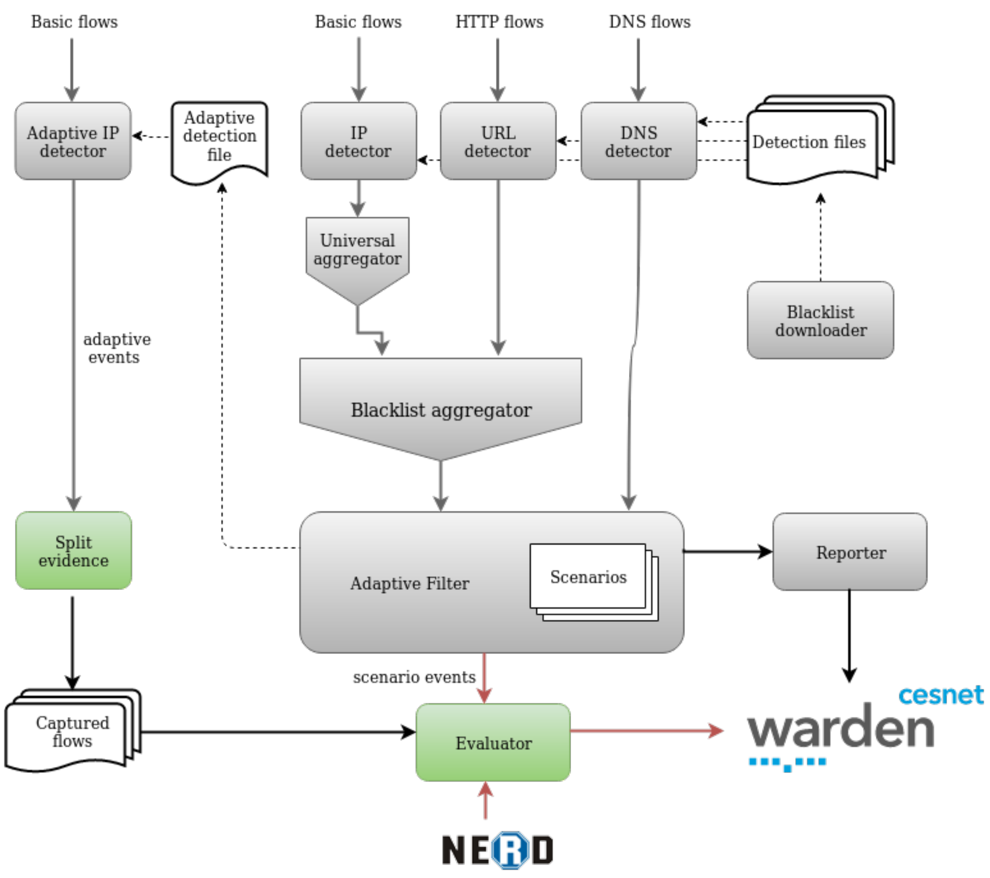

Figure 3.1: Schema of Evaluator and Split Evidence modules

Figure 3.1 illustrates the updated scheme of the Blacklistfilter module set and its interconnection with new modules. Added modules are featured in green, the existing ones are in grey. For consistency, the same labelling as in [7] is used, i.e., exchange of UniRec records and data in the JSON format are represented by black and red arrows respectively.

Chapter 4

Realisation

This chapter describes the details of the implementation of new modules. Our main goal was to create an efficient and robust program which could be easily integrated with the NEMEA system. In the process of realisation, we had to make a few modifications to the existing design and extend the functionality of libraries in the infrastructure.

4.1 Split Evidence

The module receives data from Adaptive IP Detector through a TRAP interface. The template of the input interface is shown in Table 4.1. The incoming record is converted from UniRec to comma-separated values format (CSV). Then the module inspects the `ADAPTIVE_IDS` field and appends the flow into a dedicated file based on its value. The field contains a randomly generated ID of a scenario detected by Adaptive Filter. The ID is shared by all flows originating from or addressed to the blacklisted IP during the evidence timeout. As a result, the data concerning individual scenarios are separated from others and their processing is much more convenient. The reason for choosing the CSV format over UniRec was its simplicity. It allowed us to process records without using additional libraries. The NEMEA System provides tools for conversion back to UniRec so other parts of the system still could work with the data.

Split evidence has two threads. The main thread calls blocking function `trap_recv` in an infinite loop. Whenever this function returns a record, split evidence converts are from binary to a text representation. Converted data are then passed to `write_to_file` method of the class `FileHandler` which manages file I/O operations. Tests revealed that naive implementation which opened a file, appended a record, and closed the file, was not effective enough. Keeping files opened was also not feasible because operating systems limit the number of simultaneously opened file descriptors. Thus, it was necessary to propose a method of utilising buffering. FileHandler contains a map from C++ Standard Template Library (STL) which uses scenario event IDs as keys. Each key is associated with a structure holding a buffer and a variable representing a timestamp of the last write. If the buffer has sufficient capacity, data are appended to it. Otherwise, the buffer is flushed into a file. The complexity of insertion to, finding and deleting from STL's map is $\mathcal{O}(\log n)$. The second thread periodically checks the time elapsed since the last write for each file. Whenever a write did not occur during a predefined limit, the corresponding buffer is flushed, and the structure is freed from memory.

The Module accepts 3 mandatory parameters:

- `-p PATH` — Path to a directory where to store files
- `-f NAME` — UniRec field with the value used for splitting
- `-i IFC_SPEC` — TRAP interface for incoming data

Table 4.1: Input UniRec template

UniRec field	Name	Description
ipaddr	DST_IP	Destination IP address
ipaddr	SRC_IP	IP address of origin
uint64	BYTES	Number of bytes sent
uint64	DST_BLACKLIST SRC_BACKLIST	Set to 999 for monitored clients to 0 otherwise
time	TIME_FIRST	Time and date of flow's start
time	TIME_LAST	Time and date of flow's end
uint32	PACKETS	Total number of packets in the flow
uint16	DST_PORT	Destination port
uint16	SRC_PORT	Source port
uint8	PROTOCOL	Protocol number
string	ADAPTIVE_IDS	ID of the associated scenario event

4.2 Changes in TRAP

As mentioned above, we decided to store data in the CSV format. This resulted in the need to make changes in the `libtrap`. The library did not support this feature natively. The NEMEA system already provides tools for conversion of UniRec to CSV but they cannot sort individual records into files dynamically based on a value of a given field. Consequently, it was necessary to call an external program every time we needed to analyse captured data. This method would not be feasible in a real-world scenario due to its inefficiency. As a result, a new API—`unirec2csvapi`—was implemented and made publicly available as a part of the NEMEA Framework. It is implemented in the C language like the rest of the library. Functions `urcsv_header` and `urcsv_record` allow users to effortlessly convert a classic UniRec template and records to their human-readable in the CSV format.

4.3 Evaluator

Evaluator was implemented in Python because we aimed for good code readability and swift development. C or C++ is certainly faster; however, the data Evaluator operates with are already stored persistently. Thus, in this case, speed is not the main factor. The module processes saved flows and determines traffic statistics for individual IP addresses. The module has 4 running threads:

- one receiving thread collecting scenario events from Adaptive Filter
- two worker threads reading and processing flow records from files
- a thread for reporting alerts

The module listens for reports about detected scenario events on an input interface. The reception indicates that all of the data are available and files may be manipulated safely. Incoming data are put inside a synchronized queue by the receiving thread. Worker threads monitor the queue and remove item-by-item as they process it. Each thread opens a file corresponding to the given event and begins to read individual records. The `DST_BLACKLIST` and `SRC_BLACKLIST` field in each record is inspected to identify monitored IPs (the value of the field will be set to 999; a constant defined by Adaptive Filter). An internal structure (class named `MonitoredClient`) with counters of individual traffic attributes is created for each of those addresses. Whenever all flows were examined, the worker moves the structure to the reporting queue. Reporting thread then decides whether to send an alert or not. In the first case, the saved data (CSV files) are deleted. Otherwise, they are moved to another location together with the report. The module must be run with these 3 parameters

- `--csv-path PATH` — path to output directory of Split Evidence
- `--evidence-path PATH` — directory for storage of unreported events and related data
- `-i IFC_SPEC` — TRAP interface for receiving scenario events from Adaptive Filter

```
{
    "ip":"1.2.3.4",
    "statistics":{
        "bytes_sent":143289496,
        "packets_sent":1162871,
        "flows_sent":15436,
        "bytes_recv":1140413592,
        "packets_recv":1042525,
        "flows_recv":10394,
        "floats_per_sec_recv":24.409661472705285,
        "floats_per_sec_sent":36.250484365276,
        "pkts_per_sec_recv":8.871585848868998,
        "pkts_per_sec_sent":6.408484933226045,
        "bytes_per_sec_recv":9704.589419577529,
        "bytes_per_sec_sent":789.656441862901,
        "bytes_per_pkt_recv":1093.8956782810965,
        "bytes_per_pkt_sent":123.22045695524267,
        "ips_contacted":2544,
        "ports":53
    },
    "nerd_info":{
        "asn":[   ],
        "bgppref":{   },
        "bl":[   ],
        "events":[
            {
                "cat":"ReconScanning",
                "date":"2019-04-10",
                "n":5,
                "node":"cz.cesnet.nemea.vportscan"
                .
                .
                .
```

Figure 4.1: Example of Evaluator's alert in JSON

Chapter **5**

Testing and Deployment

The performance of new modules had to be tested before their deployment on the collector. We focused on flow-processing capabilities (number of flows per seconds) and memory consumption. Split evidence and evaluator were tested with a traffic sample of the CESNET2 subnet. The total size of the sample was 1.17 GiB and it contained communication of 32,768 clients transferring 10,000,000 flows. The average memory usage and time taken to process the data was calculated from 20 runs. The UNIX program `time` was used for measurements. Additional tests of Split evidence were conducted to eliminate data races and to ensure correctness of memory handling. For this purpose, we used `Valgrind`[10], `AddressSanitizer`[11] and `ThreadSanitizer`[12] profiling tools. Programs were executed on a machine with the following configuration:

- Operating system: Linux Mint 19.1 Tessa
- Processor: Intel(R) Core(TM) i7-6700HQ CPU 2.60GHz
- Memory: 16GiB DDR4 2133 MHz
- Hard disk drive: 256GB M.2 SSD

During the tests, we also verified whether the modules cooperate with the rest of the existing infrastructure correctly. More precisely, we focused on the flow exchange between Adaptive IP Detector, Adaptive Filter, Split Evidence and Evaluator. The final number of flows stored for analysis was compared with the anticipated count obtained from the `unirecfilter`. This program filters UniRec records based on a defined set of rules and it is a part of the NEMEA Framework. We used the filter to extract flows connected to the host reported as suspicious and compared the results with the output of

[10]http://valgrind.org/
[11]https://github.com/google/sanitizers/wiki/AddressSanitizer
[12]https://github.com/google/sanitizers/wiki/ThreadSanitizerCppManual

```
==20381== Command: ./splitevidence -p csv/ -f ADAPTIVE_IDS -i f:10m_test.
    trapcap
==20381==
==20381==
==20381== HEAP SUMMARY:
==20381==     in use at exit: 0 bytes in 0 blocks
==20381==   total heap usage: 26,481,253 allocs, 26,481,253 frees,
    114,575,689,262 bytes allocated
==20381==
==20381== All heap blocks were freed -- no leaks are possible
==20381==
==20381== For counts of detected and suppressed errors, rerun with: -v
==20381== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Figure 5.1: Results of memory testing with Valgrind

the module set. This approach confirmed correctness of our implementation. The Table 5.1 shows results of conducted tests. Despite the flow-processing capabilities of Evaluator may seem low, we are confident its no impediment to its functionality due to the small number of suspicious activities observed in the network it operates on.

Table 5.1: Performance test results

Module	Time	Memory usage	Avg. flows per second
Split evidence	26.3 s	668.7 MiB	380966.9
Evaluator	309.5 s	76.1 MiB	32310.2

The final phase of testing was dedicated to measuring Evaluator's precision of identifying infected hosts. We configured the module to use thresholds obtained from experiments described in Section 2.2. Afterwards, we let the module analyse the traffic of bots from the CTU-13 dataset and the traffic of hosts from CESNET2 dataset which were marked as suspicious by Adaptive Filter. The subject-matters of our tests were the following:

1. How many bots would be recognised

2. The number of hosts from the CESNET network whose traffic marked as truly malicious (i.e., number of reports generated)

Evaluator correctly identified all bots from the malicious dataset. Results are presented in Table 5.2. The total number of suspicious clients in the CESNET2 dataset was 638. Among them were 106 addresses that did not send any flows. The module evaluated 351 clients from the remaining 532 as probably malicious and generated a report.

Table 5.2: Detection test results

Dataset	# of suspicious IPs	Reported as malicious	Percentage
CTU-13	35	35	100%
CESNET2	638	351	55%

Blacklistfilter modules were deployed together with Evaluator and Split Evidence on a testing collector inside the CESNET2 network. At the time of writing this thesis, we are still waiting for the evaluation of results from the real-world application.

Conclusion

The goal of this thesis was to create a module for the NEMEA system which would analyse the output of Adaptive Filter and calculate statistical information. For fulfilling this task, it was necessary to study the concept of network flows, principles of their collection, and their analysis. Afterwards, I had to acquaint myself with the design and implementation of Adaptive Filter and related modules from the NEMEA framework. In the course of writing this work, a unique dataset was created. It represents normal traffic observed in the CESNET2 network. Statistical analysis of the dataset and samples of various malware led to the identification of several characteristics specific for botnet traffic. These results could be used to increase the precision of threat detection.

Consequently, it should help security teams in their task of recognizing and responding to threats. Our experiments and results were summarized in an article submitted for the prestigious ACM Internet Measurement Conference. At the time of writing this work, we are waiting for its acceptance.

Our effort led to the creation of a new module for NEMEA system—Evaluator. The module cooperates with Adaptive Filter and could be integrated into the existing infrastructure. Tests in a closed environment proved the correctness of Evaluator's design and confirmed it could be deployed in the real-world scenario. Therefore, the main goals of the thesis have been fulfilled.

Bibliography

1. FELDMANN, A.; GREENBERG, A.; LUND, C.; REINGOLD, N.; REXFORD, J.; TRUE, F. Deriving traffic demands for operational IP networks: methodology and experience. *IEEE/ACM Transactions on Networking* [online]. 2001, vol. 9, no. 3, pp. 265–279 [visited on 2019-03-10]. Available from DOI: `10.1109/90.929850`.
2. AITKEN, P. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information* [online]. RFC Editor, 2013 [visited on 2019-04-19]. ISSN 2070-1721. Available from DOI: `10.17487/rfc7011`. Technical report.
3. CLAISE, B. (ed.). *Cisco Systems NetFlow Services Export Version 9* [online]. RFC Editor, 2004 [visited on 2019-04-19]. ISSN 2070-1721. Available from DOI: `10.17487/rfc3954`. Technical report.
4. CEJKA, Tomas; BARTOŠ, Václav; SVEPES, Marek; ROSA, Zdenek; KUBATOVA, Hana. NEMEA: A Framework for Network Traffic Analysis [online]. 2016 [visited on 2019-04-28]. Available from DOI: `10.1109/CNSM.2016.7818417`.
5. BARTOS, V; ZADNIK, M; CEJKA, T. Nemea: Framework for stream-wise analysis of network traffic. *CESNET, ale, Tech. Rep.* [online]. 2013 [visited on 2019-04-03]. Available from: `https://www.cesnet.cz/wp-content/uploads/2014/02/trapnemea.pdf`.
6. BARTOŠ, Václav. Creating a Network Reputation Database [online]. 2016 [visited on 2019-05-07]. Available from: `https://nerd.cesnet.cz/tnc16-poster.pdf`.
7. ŠUSTER, Filip. *Automatická detekce podezřelého síťového provozu pomocí blacklistů*. Praha, 2019. Available also from: `https://dspace.cvut.cz/handle/10467/79796`. Master's thesis. České vysoké učení technické v Praze. Fakulta informačních technologií.

8. FAN, Jinliang; XU, Jun; AMMAR, Mostafa H; MOON, Sue B. Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme. *Computer Networks*. 2004, vol. 46, no. 2, pp. 253–272.

9. GARCÍA, S.; GRILL, M.; STIBOREK, J.; ZUNINO, A. An empirical comparison of botnet detection methods. *Computers & Security* [online]. 2014, vol. 45, pp. 100–123 [visited on 2019-03-24]. Available from DOI: `10.1016/j.cose.2014.05.011`.

10. RSTUDIO TEAM. *RStudio: Integrated Development Environment for R. Version 1.2.1335* [software]. Boston, MA, 2018 [visited on 2019-03-27]. Available from: `https://www.rstudio.com/`.

Appendix A

Acronyms

CSV Comma-separated values

DNS Domain Name System

DPI Deep packet inspection

IoT Internet of Things

IP Internet Protocol

JSON JavaScript Object Notation

STL Standard Template Library

URL Uniform Resource Locator

TOS Type of Service

TTL Time To Live

VM Virtual Machine

Appendix B

Contents of enclosed CD

```
readme.txt.........................the file with CD contents description
src.......................................the directory of source codes
  impl.........................................implementation sources
  thesis...............the directory of LaTeX source codes of the thesis
text...........................................the thesis text directory
  thesis.pdf.............................the thesis text in PDF format
```