UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

PH.D. THESIS IN

INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

BIG DATA ANALYTICS FOR FLOW-BASED ANOMALY DETECTION IN HIGH-SPEED NETWORKS

MAURO GAROFALO

TUTOR: PROF. GIORGIO VENTRE

XXIX CICLO

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE

Abstract

The Cisco VNI Complete Forecast Highlights clearly states that the Internet traffic is growing in three different directions, Volume, Velocity, and Variety, bringing computer network into the big data era.

At the same time, sophisticated network attacks are growing exponentially. Such growth making the existing signature-based security tools, like firewall and traditional intrusion detection systems, ineffective against new kind of attacks or variations of known attacks.

In this dissertation, we propose an unsupervised method for network anomaly detection. This method is able to detect unknown and new malicious activities in high-speed network traffic.

Our method uses an innovative detection algorithm able to identify the hosts responsible for anomalous flows by using a new statistical feature related to traffic flow. This feature is defined as the ratio between the number of flows generated by a host and the number of flows it receives.

We evaluate our method with real backbone traffic traces from the Measurement and Analysis on the WIDE Internet (MAWI) archive. Fur-

thermore, we compare the results of our method with MAWILab archive, a database that assists researchers to evaluate their traffic anomaly detection methods.

The results point out that our method achieves an average positive prediction rate (i.e. Precision) of 90% outperforming the four MAWILab detection methods in terms of false negative rate.

We deploy three cluster configurations to evaluate the horizontal and vertical scalability performance of the proposed architecture and our method shows outstanding performance in terms of response time.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

This chapter gives an overall description of this dissertation. It is divided into four sections. First of all, Section 1.1 describes the context in which the thesis has been developed. Section 1.2 presents the motivation. Then, Section 1.3 shows the primary objectives and the original contribution. Finally, Section 1.4 outlines the remaining of the thesis chapters.

1.1 Background

Nowadays, more and more services are available on the Internet, and their use becomes widely pervasive. Thus most of such services are heavily dependent on a network access and their use is no longer confined to IT experts. People use chat applications and social media to communicate with each other, e-banking services to manage their money and watch movies on video streaming services. It has been reported that Video

streaming is responsible for about the 70% of Internet traffic peak in North American fixed access networks, and this figure is expected to reach 80% by 2020 [17].

Apart from the ascertained foreseen increase of data Volume for the Internet traffic also Velocity and Variety of data will increase in the next few years.

This trend is driven mainly by two factors. In the first place, the steady growth of devices connected to the Internet. In fact, a large variety of devices, such as mobile phones, personal computers, IP cameras, wireless measurement sensors and more in general Internet of Things [46, 32], leads to the creation of a significant data volume in a variety of data types. Second, the improvements in the communication technology allowed the spreading of the current high-speed networks with traffic speed never seen before. Since Volume, Variety, and Velocity are the 3Vs characteristics that define the Big Data [53], thus the network traffic monitoring can be considered as a Big Data problem [28]. In the meanwhile, the number of attacks against computer networks is increasing at the same notable rate. In particular, the homogenization of mobile operating systems, where Android take the 87% of market share [39], combined with the increase in computing performance of smartphone devices and the spreading of 4G communication technology [67], make the smartphones a desirable target for attackers. In this scenario, measuring traffic characteristics in a more effective way is crucial for different network activities such as network planning, traffic management and network security. A primitive way to make the network less vulnerable from potential attacks is the use of a firewall. A firewall is often deployed between the *trusted* local network and *untrusted* networks in order to provides a filter for incoming

and outgoing traffic. It stops unauthorized access through the use of simple rules specified by the network administrator. A firewall can provide basic preemptive protection for the systems but it can not detect intrusion or more sophisticated attacks. For this purpose, network administrators need more advanced tools able to identify potential threats to the network. Intrusion Detection System (IDS) is a software (or hardware) alarm-system that looks for intruders who have overcome the measures previously adopted. Furthermore, due to the rapid evolution of attack techniques it is crucial to use detection methods which are effective against both new kinds of attack and variations of known ones. A typical signature-based IDS may not be sufficient to detect these kinds of attack because they need a constant updating of new accurate attack signature. Anomaly-based IDS (ADS) are designed to overcome this issue. In fact, ADSs are able to detect such kind of attacks, but even if attacks are identified, existing methods are usually too slow in terms of response time.

In this dissertation, we propose a network monitoring architecture for anomaly detection (AD). We are capable to achieve real-time like response time by introducing an innovative AD method which exploits Big Data Analytics framework. Finally, we provide a performance and scalability analysis of this architecture in the case it is deployed on an in-house cluster or in the cloud. Moreover, we show challenges and open issues in anomaly detection, and the existing network anomaly detection methods, including their advantages and disadvantages, are discussed.

1.2 Motivation

Anomaly detection is a critical problem common to many research domain. An anomaly can be defined as a pattern in the data that does not conform to the expected behavior, and an Anomaly Detection System (ADS) analyzes the input dataset with the aim of detecting these deviations from the standard pattern of behavior. In the computer network, anomalies are unusual and significant changes in network traffic, and the development of ADSs in high-speed networks come with many challenges to face. The main of them are the lacks of ground truth (Section 3.5), the high false alarm rates, the route cause identification, a significant amount of data to be processed, the time-spending computation, and the needed of a real-time response. With the aim to cope these challenges in mind, this dissertation proposes a flow-based anomaly detection system using Big Data Analytics. As seen in Section 1.1, systems for monitoring high-speed network traffic involve a huge amount data to be observed and its analysis can be prohibitive. Analyze traffic data at packet-level can be not only time-consuming but also superfluous. In order to reduce the volume of data to be analyzed, our proposed architecture analyzes the traffic data at flows-level rather than at the packet-level. Beside the 5-tuple (i.e. source and destination IP addresses, source and destination port, and transport protocol) that identify a flow, for each of them, we record for each flow record a set of statistical features, namely the number of bytes sent, the number of packets, and the duration. For our anomaly detection algorithm. We preferred to use an unsupervised method because supervised approaches assume the existence of a training set (i.e. ground truth). A ground truth dataset is composed by a number of labeled samples, where the labels represent information about at which class (such as

normal or anomaly activity) the entry belongs. Due to the lack of high-quality labeled traffic, we have chosen a statistical-based approach rather than signatures or patterns-based methods. In fact, this approach can operate without any previous knowledge about the traffic. From Telco Operator point of view, an anomaly justifies a more in-depth analysis only if the amount of traffic involved can be deleterious to the proper operation of its network infrastructure. Accordingly, we have focused our efforts towards the identification of volume anomalies which involve huge amounts of NetFlow in a short time. Finally, to realize a system able to detect anomalies in real-time, a scalable and efficient solution is still needed, and traditional systems for data management and analysis are ineffective for the purpose. On the other hand, we exploit modern big data analytics (BDA) tools (i.e. Hadoop, Apache Spark, and Kafka) that are able to handle these time-spending tasks in a more efficient way compared to traditional approaches.

1.3 Thesis Contribution

In this thesis, we propose a system to deal volumetric anomalies (such as DoS, DDoS or port scan), in a Tera-bps networks. Due to the data size and velocity, we analyze traffic flows instead of the single packets passing through the network to face the amount of data expected, and exploit a Big Data Analytics framework (i.e. Apache Spark) to provide a scalable implementation of the anomaly detector.

Summarizing, the contributions of this thesis are as follows:

- The first contribution is the introduction of a volume-based algorithm for network anomaly detection. This algorithm analyzes all the IP flows traversing the monitored network instead of inspecting the payload of each packet. The algorithm makes use of the ratio of the number of flows sent and the number of flows received by every host on the monitored network identifying which is responsible for the anomalies. This method does not require prior knowledge, such as a training dataset or a database of attacks behaviors. We create the input dataset transforming traces from MAWI archive into network flow traces. To evaluate the accuracy of our method we use the MAWILab ground truth, a labeled dataset of identified anomalies present in the MAWI archive traces.

- The second contribution is the development of an architecture that uses algorithm mentioned above to detect the anomalies in real backbone traces. In particular, this architecture has three targets, collect the flows coming from the probes (i.e. flow exporters) deployed around the monitored network, process the flows collected and finally analyze the flows to detect anomalies in the traffic. This architecture exploits Apache Spark, a Big Data Analytics framework, to improve the processing performance.

- The third contribution is the deployment of the anomaly detection architecture on the cloud. We implement a testbed on AWS in order to evaluate how our method performs when scaling up and out.

1.4 Thesis Organization

The reminder of this thesis is organized as follows:

- **Chapter 2** (Network Traffic Monitoring),presents an overview of network traffic measurements techniques. Furthermore, a description of active and passive measurements and their applications is provided.
- **Chapter 3** (Intrusion Detection) presents intrusion detection concepts and reviews related work in the area of the network anomaly detection. Firstly, a general overview of IDSs taxonomy is provided. Then, we introduce the definition of attack and anomaly, and finally, a description of the most spread network anomalies is provided.
- **Chapter 4** (Big Data Analytics and Network Monitoring) provides an introduction to the background of Big Data in general. Then insights on some application of Big Data Analytics technologies to network management, and network architecture solutions for Big Data processing applications like Hadoop. Finally, a brief overview on Big Data analytics tools concludes the chapter.
- **In Chapter 5** (An architecture for anomaly detection) the proposed architecture and the anomaly detection method are presented.
- **Chapter 6** (Evaluation and Results) describes the tests performed and it presents the results of these tests. an evaluation
- Finally, **Chapter 7** concludes this dissertation by summarizing my contributions and presents some ideas for the future work.

Chapter 2

Network Traffic Measurements

In this chapter we describe the network flow monitoring technique, as flows were chosen as data source for the proposed method. The chapter starts with Section 2.1 where a general description and comparison of two main network traffic measurements methods, namely active measurement and passive measurements is proposed. Then, in Section 2.2, some details related to packet level as well as flow level passive measurement techniques are presented.

Finally, in Section 2.3. the classic flow monitoring architecture with its two main components, the flow exporter and the flow collector, are described.

2.1 Network Monitoring

Network monitoring aims to infer useful traffic information through the monitoring, collection, and analysis of traffic data. In order to monitor the network performance, two kinds of measurement can be used, namely active and passive measurement methods [58].

2.1.1 Active Measurements

Active methods are based on the observation of a packets stream [88] created ad hoc for the purpose of measuring service level parameters such as the delay, the packet loss, and the jitter [75]. Active measures involve probes that generate test traffic as similar as the traffic that would be generated by a legitimate user of the service. Synthetic traffic is injected into the network to simulate the use of the service by the user. This traffic is received by one or more intermediate probes. In order to facilitate the measurement, one or more packet fields (such as timestamps or sequence numbers) are modified. Once an intermediate probe receives the test traffic, it can have two behaviors. If the parameter to be estimated involves only the outward traffic, it registers the observed results and evaluates the performance. Otherwise, if the parameters to be evaluated also depend on the return traffic, it reflects the traffic received to the sending probe and allows that it evaluates the performance parameters. Traditional tools for this type of measures are ping [Muuss], and traceroute [52]. The first uses the ICMP (Internet Control Message Protocol) [69] echo request to measure the network latency between two hosts (i.e. the round-trip time). The latter is used for discovery the routing paths from the source host to the destination target host. Usually, it is implemented by transmitting a series

of probe packets with increasing time-to-live values. Each hop in a path to the destination host rejects the probe packet (probe's TTL too small) until its time-to-live becomes large enough for the probe to be forwarded. Each hop in a traceroute path returns an ICMP message that is used to discover the hop and to calculate a round-trip time [70]. Active methods are useful for making measurements in case of the traffic inspection is impossible in a passive way. On the other hand, Active measurements techniques, by injecting additional traffic to the network can potentially influence the measured properties. Thus, these measurements methods need to handle their effects on the measurements quantifying their effects to the measures and implementing practices to minimize such effects. An application of active measurements is presented in our paper [8]. In this work, we proposed an active measurement methodology that allows to acquire and analyze performance data and topology information about the infrastructure of video hosting providers. In the following sections, as in the rest of this dissertation, we focus on the problem of analyzing the network traffic traces collected using passive measures.

2.1.2 Passive Measurements

In contrast to active measurements, passive measurement methods are non-intrusive. Passive methods gather network information by observing the traffic stream of interest without remove, change or add any packets to the packet stream [88]. The functioning of these methods depends on the use of one or more observation points [88] located inside the monitored network (i.e. on switches, routers, or gateways). According to the method in which the observation points collect the data, passive measurement techniques can be classified into two type. The first type of passive measurement

inspects into the packets and receives all packets that pass through the observation points. The second technique filters the traffic of interest and collects only the packets that matching the filter criteria in order to limit the amount of traffic to be analyzed. Regardless the used technique, Observation Points have to send the observed packets, the traffic statistics or both to a collector for the furthermore analysis. In fact, this information is used for various purposes, for example, to infer performance metrics or unveil particular user behaviors. Thus, in network configurations with high data rate, the traffic load generated by the communication between Observation Points and collectors may itself influence the measurements.

2.2 Packet-Level vs. Flow-Level Measurements

The passive measurement can gather the observed packet stream at mainly two different levels of granularity: packet-level, and flow-level.

2.2.1 Packet-Level

In the first case, the measurements are performed using the information related to the single packets. Both packet capturing and analysis phases can be performed inside the Observation Points, or alternatively, the packets are streamed to a remote system for the analysis phase. With the increase of data rate, and consequently with the number of packets, packet-level analysis and storage phases become more and more resources demanding. To cope this problem, the measurements can be performed on a restricted subset of packets of interest by filter or sampling algorithms. Packets sampling can be achieved by random algorithms where the packets selec-

tion depends on a random process and systematic algorithms which aim at identifying a statistically significant subset of packets. IETF standard proposals for techniques for packet sampling and filtering are presented in [88] and [19].

2.2.2 Flow-Level

On the contrary respect to the aforementioned packet-level measurements, flow-level measurements collect the traffic information related to a packet aggregation, the flow. The flow represents a statistic summarization of network traffic. According to Internet Engineering Task Force (IETF), a flow is defined as "a set of packets or frames passing an Observation Point in the network during a certain time interval. All packets belonging to a particular flow have a set of common properties" [20]. Packets belonging to a flow have in common a set of key properties. Traditionally, these keys, also know as 5-tuple, are the source IP address, the destination IP address, source and destination transport ports and transport protocol. Therefore, a 5-tuple aggregates all packets belonging to a one-way communication between two hosts on a single socket. In flow-level measurements, the observation points are also called Flow Exporters. For every flow, the Flow Exporter generates various traffic statistics, and it stores them in a flow record. Usually, Flow Exporters are used only for the monitoring phase. Subsequently, they "export" the flow records to a remote component called Flow Collector. Flow exportation is triggered by an active or a passive timeout. In the first case, long-lasting flow records are exported when the timeout expires. On the other hand, in passive timeout, the flow exportation is triggered after a defined period where no new packet is observed. As

well as in packet-level traffic measurements, the demand for computing resources increases hand in hand with the growth of packet rate.

2.3 Flow Monitoring

Flow monitoring architecture consists of a hierarchy of functional components. Features and responsibility of this architecture are spread across these components that perform their tasks subsequently and independently, in a perspective of scalability of the system. Figure 2.1 shows a typical architecture for flow-level network monitoring. Usually, a flow monitoring architecture is composed by a set of Observation Points, where the packets are captured, one or more flow exporter and metering components, and one or more flow collectors. Finally, the captured data are available to a monitoring or an analytics framework for on-line analysis, forensics analysis or both.

2.3.1 Flow Exporter

The flow exporter is the component responsible for the packet observation, the metering process, and the flow exporting process. Observation Point is defined as a monitored network spot where the packets stream is observed. The ports of a router, the interface of a packet forwarding device or the line to which a probe is attached are examples of Observation Points. Flow Exporter exploits a set of Observation Points to read the packets from the line and extracts the header information of each packet passing through them. The final step of packet observation phase is packet filtering and sampling. Packet filtering and sampling algorithms aim to select a

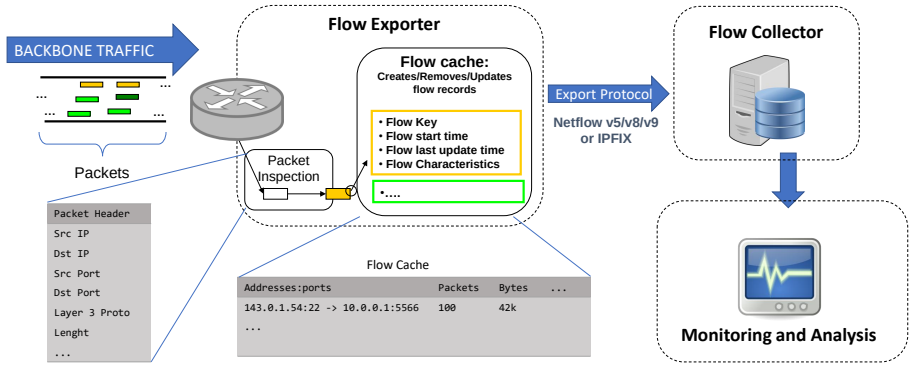

Fig. 2.1 Overview of a typical Flow Monitoring Architecture

certain packets subset for the following phases. The reduction of data to be processed plays a crucial role in using less memory resource, and to preserve the bandwidth. In flow metering stage, the flow exporter aggregates the packets according to a set of key properties (i.e. the flow 5-tuple), into a flow record. All active flow records are stored in a table called flow cache. For each packet, the flow exporter compares the incoming packet header information with the flow entries in the flow cache. If the header information does not match with any existing flows, a new entry is created in the flow cache table. Otherwise, the corresponding flow entry is updated with the information of the new packet header. Flow Exporter looks for expired flow entries in the flow cache. When flow records expire, the flow exported forwards (i.e. *exports*) them to the flow collector. Flow cache entries are considered as expired when they meet one of the following triggering conditions:

- The flow cache memory is full. All the flow entries in the cache table are considered as expired.
- The flow entry exceeds its active timeout. Long-lasting flows active for more in which a specified time are considered as expired.
- The flow entry exceeds its passive timeout. After a specified time that no packets belonging to the flow have been observed for it, the flow is considered as expired.
- FIN or RST flag is observed for a TCP flow. When a TCP packet with a FIN or RST flag has been observed for a flow, the flow is considered as expired.

An example of the fields of a Netflow Version 5 flow record is summarized in Table 2.1. Cisco Netflow version 9 [18] and the IPFIX [20] are at state of the art in exporting flow records protocols.

Table 2.1 Flow record fields of a Cisco Netflow Version 5

Bytes	Contents	Description
0-3	srcaddr	Source IP address
4-7	dstaddr	Destination IP address
8-11	nexthop	IP address of next hop router
12-13	input	SNMP index of input interface
14-15	output	SNMP index of output interface
16-19	dPkts	Packets in the flow
20-23	dOctects	Total number of Layer 3 bytes in the packets of the flow
24-27	First	SysUptime at start of flow
28-31	Last	SysUptime at the time the last packet of the flow was received
32-33	srcport	TCP/UDP source port number or equivalent
34-35	dstport	TCP/UDP destination port number or equivalent
36	pad1	Unused (zero) bytes
37	tcp_flags	Cumulative OR of TCP flags
38	port	IP protocol type (for example, TCP = 6; UDP = 17)
39	tos	IP type of service (ToS)
40-41	src_as	Autonomous system number of the source, either origin or peer
42-43	dst_as	Autonomous system number of the destination, either origin or peer
44	src_mask	Source address prefix mask bits
45	dst_mask	Destination address prefix mask bits
46-47	pad2	Unused (zero) bytes

Filtering and Sampling

Before the flow exporting process, filtering and sampling algorithms can be performed to select a subset of flow records. On the one hand, the goal of these phases is to reduce the resources demand of all subsequent steps, which is the same goal of filtering and sampling at the packet level. On the other hand, these two processes are executed on the flow entries in the flow cache instead of the single packet. Thus, when a flow matches any filtering or sampling criteria, either all the packets belonging the flow are considered matching or none.

2.3.2 Flow Collector

The Flow Collector is the component responsible for retrieving and storing of the flow records from the Flow Exporters in the network. At this stage, pre-processing activities are performed for helping further monitoring and analysis process. Examples of pre-processing tasks performed by this component are data anonymization, data compressing, data aggregation, and filtering.

Chapter 3

Intrusion Detection System

This chapter reviews related work in the area of network anomaly detection. The chapter starts with a general overview of the Intrusion Detection Systems. Then, we presented more details on the anomaly detection techniques closely related to the approach proposed in this Dissertation. Finally, some remarks on existing datasets for evaluating network anomaly detection systems are given.

3.1 Introduction

Intrusion detection system (IDS) is an invaluable network security tool that aims to detect potential malicious activities against information systems, and recording useful information for further forensic investigations. IDS is a component in hardware or software that monitors the actions taken in a specific environment (e.g. the host, the network or both) and analyzes

them in order to determine if these activities are a legitimate use of the environment or the evidence of intrusions or attacks against the system. IDS does not prevent the attack from reaching the different resources, either it directly analyzes the real traffic from the client to the server and vice-versa. On the contrary, it analyzes a copy of the traffic that pass through the network probes, such as the routers, computers, etc.

IDS can be classified according to the place where they are deployed in *Host-based*, and *Network-based* [25], [74].

3.1.1 Host-Based IDS

Host-based IDS (HIDS) aims to disclose attacks targeted to a specific system (also called host). HIDS running on the monitored host depends on its architecture and operating system. It monitors and analyzes the inbound and outbound traffic only from the host and processes high-level information such as system calls, commands running on the host, changes in specified files. Finally, it reports an alert when it detects a deviation from normal behavior. An HIDS is effective in detecting intrusions that occur within the local network (e.g. a virus installed on the monitored host or if the attacker is on the same network as the victim).

3.1.2 Network-Based IDS

A Network-Based IDS (NIDS) aims to reveal attacks or intrusions on a network as a whole, and its functioning is independent of the architecture of the hosts which may be different from each other and belongs to the monitored network. The primary purpose of these intrusions is due to

the attacks launched by hackers outside the network who want to obtain unauthorized access to the system for information stealing or to disrupt the normal operation of the network. NIDS is deployed at strategic points within the network to analyze traffic exchanged to and from all hosts within the network itself. NIDS is effective in detecting intrusions from outside the monitored network, such as Denial-of-Service (DoS) attacks, botnet, and network scans.

3.2 IDS Taxonomy

According to the approach used to identify the intrusions, IDSs can be classified in misuse-based and anomaly-based.

3.2.1 Misuse-Based IDS

A *misuse-based IDSs* detects intrusions by matching current pattern with a set of events, sequences of events (rules) that are the symptom of a security threat. These rules are contained in a predefined knowledge base of attack behaviors. This kind of IDS has a high accuracy to detect well-known attacks, but it is unable to recognize new or variants of known attacks.

3.2.2 Anomaly-Based IDS

On the other hand, *anomaly-based IDSs* do not need attack signature because they perform a comparison with the regular traffic pattern analyzing the network traffic to find abnormal activities, without distinguishing from

malicious and benign. The principal research challenges in ADS are the high number of false alarms rate, the lack of publicly available labeled datasets to be used as ground truth for network anomaly detection[2]. As the network is constantly evolving, the model of normal behaviors may not be accurate in the future. Even if ADSs generate a higher false alarms rate than misuse-based IDSs, they come with the invaluable advantage of being able to detect unknown novel attacks.

3.3 Network Activities

Network activity can be classified according to the purposes of the actors who perform in two types, legitimate (or benign) and malevolent.

3.3.1 Benign Activities

The Benign activities are all the activity that a user or a device performs using the network according to the intended purpose by the network administrator. For example, a student who checks e-mail through the network of the university campus, or a host using a DNS service to resolve the URL of a website perform legitimate activities.

3.3.2 Malicious Activities

On the contrary, a malicious activity is an action carried out by an attacker or a compromised host that has the sole purpose to damage of the services and the hosts on the network. This type of activities can be designed to

capture, destroy, modify, or access to information from a network, a host, or users without permission. For example, a bot (i.e a compromised host), which scans a network looking for vulnerable hosts, or a group of hosts that makes a huge number of requests to a service with the only purpose of making it inaccessible to the legitimate users perform malicious activities.

3.3.3 Network Attacks

An attack is defined as an intentional act by which an entity attempts to evade security services and violate the security policy of a system [76]. According to its purpose, an attack can be classified in passive or active attacks.

- Active attacks attempt to alter system resources or affect their operations.
- Passive Attacks try to get information on the target system without changing its resources and subsequently exploit these data to prepare an active attack on the target system.

An attacks Taxonomy was provided by [37].

3.4 Network Anomalies

Network anomalies could be generated either by non-malicious usage of the network and network attacks. In the following, we describe more in these anomalies type.

3.4.1 Alpha flows

Alpha flows are anomalies where the traffic increases from just a few high-volume connections between two hosts. Usually, alpha flows are caused by the transmission of high dimension file over high-bandwidth links.

3.4.2 Flash crowd

Flash crowd consists of an unusually large and quick demand of a specific resource from many clients (i.e. the downloading of a security patch or the live video streaming of a wide interest event). These kinds of events lead the increase of both inbound and outbound traffic (i.e. requests/responses) from the server that contains the resource.

3.4.3 Worms

Worms [71] are self-replicating malicious software that tries to infect other hosts by exploiting specific vulnerabilities. During the propagation phase, the infected hosts send a small number of packets to a large number of target hosts on the Internet.

3.4.4 Network Scans

Network Scan, also known as horizontal scan, is a malicious activity that aims to identify which hosts belonging a specific network (i.e. the target) are alive. To perform a Network Scan, a single host sends a huge number of

probe packets to a wide range of destination hosts (i.e. target). Depending on its purpose, an attacker may send each probe packet to a large set of ports or a specific one. In the first case, the attacker is interested in discovering which of the contacted hosts are active. In the second one, they can also detect if the target host has a specific service active on it. The most common scans use probe packets with SYN or ACK TCP flag active, namely SYN scans and ACK scans. Other TCP flags, such as invalid combination for the protocol, can be used.

3.4.5 Port Scans

Port Scan, also know as vertical scan, is a malicious activity that aims to detect all service running on a specific host. Essentially, this type of scan involves a single host that sends the probe packets to a wide range of ports of the target. This operation is performed to check which services are available on the victim host. Usually, this kind of scan uses packets as small as possible for efficiency purpose, such as UDP packet with a one-byte payload.

3.4.6 DoS attacks

DoS attack exploits known vulnerabilities of the communication protocols in order to incapacitate the target host ability to respond to a legitimate request. This malicious activity is performed sending a huge number of requests to the victim server. This overload of requests slows the server in quickly responding to a legitimate user until the target host is unable

to handle all requests both from attackers and legitimate users. Common DoS attacks are TCP SYN Flooding, UDP Flooding, and ICMP Flooding.

TCP SYN Flood

TCP SYN Flood is a DOS attack that takes advantage of the vulnerabilities of the TCP three-way handshake. SYN flood attack aims to occupy all the memory resources of a victim by sending a disproportionate number of TCP SYN requests with spoofed source IP. The victim responds with SYN-ACK packets, but do not get answers because its source IP is forged. Because of the incomplete three-way handshakes, the memory resources of the victim host tend to run out (this depends on the operating system used by the victim), and as a result, the victim is no longer able to accept the legitimate connection requests for a while.

ICMP Flood

ICMP Flood is a flooding attack similar to the SYN flood. It occurs when an attacker overloads his victim with a huge number of ICMP echo requests with spoofed source IP. This type of attack still causes many hardships for network administrators.

3.4.7 DDoS Attacks

Another widely used type of attack is the Distributed DoS (DDoS). It has the same goal of what Dos, however, involves more hosts to give more power to his attack. In DDoS attack, the attacker sends a huge number of

packets from multiple hosts toward the victim network. Despite Dos and DDoS threats are not next-generation kinds of attack, they are constantly changing and tend to be a serious threat not only to the users but also for the network infrastructure also.

3.5 Datasets

One of the primary challenge that the anomaly detection poses is the lack of labeled ground truth for evaluating the detection methods. Ground truth data are tough to acquire since they require a solid domain knowledge, may have privacy issue (in the case in which real traffic traces). Last but not less, a high-quality ground truth dataset often needs to be manually created which is a very time-spending process. In the following, useful methods to create a ground truth are presented. Moreover, we provide a brief overview of the available datasets mainly used in literature.

3.5.1 Synthesized traces

One common approach to evaluate a detection method is to use a dataset where the anomalies are synthesized. These anomalies, combined in a background trace (i.e. normal traffic), compose the dataset utilized in the evaluation phase. The background trace can be both real traffic or synthetic traffic. On the one hand, this approach has the invaluable advantage of fully controlling the anomalies inside the dataset. In fact, all the features of each anomaly are defined, such as the duration, start time and end time, the number of packets and their rate, the number of contacted IPs. On the other hand, these datasets have the disadvantage that the anomalies

synthesized refer to well-known behaviors and no recent attacks are being available. Moreover, the distribution of normal traffic and anomalous events is quite unrealistic. Notwithstanding these issues, these kinds of datasets are still spreadly used by researchers in recent work for mainly two reasons: the datasets are publicly available, and they are the unique tool to strictly compare their detection methods with the ones existing in the literature. In the following, we introduce the most important available datasets of synthesized anomalies.

DARPA

The DARPA (US Defense Advanced Research Projects Agency) dataset is one of the most widely used synthetic dataset for anomaly detection evaluation. DARPA 1998/99 dataset was prepared and managed by MIT Lincoln Labs[51]. Its objective was to evaluate intrusion detection systems. The dataset provides five weeks of simulated network traffic collected from the US Air Force-based network and the Internet. More specifically, the traffic data from weeks one and three are anomaly free. The data from weeks two, four, and five contain both normal and simulated attack traffic. Attacks were injected into the synthetic background traffic, using real attack tools in an isolated test bed. The simulated attacks consist of 177 instances of 59 different types of attacks, such as DoS, U2R, R2L, and probe attacks.

KDD99

The KDD99 [KDD] dataset was created by processing the 1998 DARPA Intrusion Detection System Evaluation tcpdump traces. The result is a

dataset consisting of nearly five thousand records where each record represents 41 traffic features extracted from labeled connection records. The traffic connection records are labeled, as in DARPA dataset, as normal, DoS, R2L, URL or probe attacks. The KDD Cup dataset provides both training and test set. The KDD set is about 20 years old and has not been updated. Therefore, this dataset may not be a good choice for the evaluation of anomaly detection methods that aim to detect current anomalies in nowadays traffic. However, this dataset is still one of the most popular for researchers who want to evaluate their data mining-based methods.

3.5.2 Collected Real Traffic

The other approach to evaluating and validate an anomaly detection method consists in using datasets of traffic collected from real-world networks, such as backbone, campus, and enterprise networks. Using real traffic leads the advantage that anomaly detection methods are "certified" to being able to identify real abnormal events in real network traffic. However, captured datasets are often privacy sensitive, especially when they contain the packet payloads and IP addresses.

CAIDA

CAIDA [CAIDA], the Center for Applied Internet Data Analysis is a collaborative undertaking among organizations in the commercial, government, and research sectors aimed at promoting greater cooperation in the engineering and maintenance of a robust, scalable global Internet infrastructure. It collects several different types of data at geographically and topologically diverse locations and makes this data available to the

research community to the extent possible while preserving the privacy of individuals and organizations who donate data or network access. CAIDA provides several traffic datasets for different research purposes such as anonymized backbone traffic, DDoS attack dataset or telescope datasets used for the observation of anomalous behavior. Since these datasets are composed of raw traffic traces that have no labels available, they are unusable as validation set of detection methods.

Abilene

Abilene [Abilene] provides another public available real traffic dataset that is widely used in anomaly detection research domain. The Abilene (also knows as Internet2) is a backbone network that connects various US campus and peering with numerous European and Asian research networks which include 11 Point of Presence (PoP). For each Origin-Destination Flows, with a total of 121 OD flows, this dataset provides the flow statistics measured at five minutes intervals. These statistics are available in Matlab files.

MAWI

The MAWI [16] dataset is a freely accessible real backbone network traffic dataset. The main advantage of this dataset compared to the previous ones is twofold. Firstly the collected traces are very up-to-date. Secondly, MAWI group provides a dataset with labeled anomalies (i.e. MAWILab) for every instance in the traces mentioned above. A more detailed description of MAWI and MAWILab datasets will be provided in chapter 6.

3.6 Anomaly Detection

Anomaly detection is a critical problem common to many research domain. An Anomaly Detection System (ADS) analyzes the characteristics of the input data with the aim of discovering *deviations* from a normal pattern of behavior. These deviations are also referred to as anomalies. The term *anomaly* often referred as different names such as outliers, exceptions, aberrations, etc., refers to patterns behavior that differs from what we would normally expect. In the computer network, an anomaly may be caused both by an attack, and a non-malicious traffic activity. According to the IETF definition, "an attack is a deliberate activity by which an entity attempts to evade security services and violate the security policy of a system" [76]. Due to the evolution of the attack methodologies, it is important to use methods able to detect the malicious activity of any nature, both known and unknown. This could be done through an anomaly intrusion detection system. Anomaly IDS approach was introduced for the first time by Denning [26]. Many surveys have been published to classify anomaly detection systems according to the detection methods. The authors in [36], [14] provide a comprehensive review of outliers detection methodologies. They identify three fundamental approaches to the outliers detection problem:

- *Unsupervised* - This approach operates without an *a priori* knowledge of data. It is based on the assumption that in the dataset the normality events are statistically more frequent than anomalies.
- *Supervised* - This approach uses the models of the normality and the abnormality to operate. In other words, a training set of labeled instance of both normal and anomalous events is needed.

- *Semi-supervised* - This approach uses models normality. In this case, the training set contains only the instance labeled for the normal events.

It uses two technique: Statistical calculations or Machine Learning (ML) algorithms.

3.6.1 Statistical Methods

Anomaly detection systems based on statistical methods consist of two phases. First, the system observes and collects one or more statistical features of network traffic, subsequent it compares the current state with the stored one using a stochastic method to detect behavior changes. One critical of the challenges in the development of resilient and secure systems is the efficient detection of the malware. In fact, malware is often the point of beginning for various kinds of attacks, such as Distributed Denial of Service (DDoS), phishing and email spamming, which are performed through bots deployed on the target hosts. With the purpose to face this challenge, [55] have presented an anomaly detection technique, that uses an Ensemble Empirical Mode Decomposition (E-EMD) algorithm in order to conduct a statistical characterization and decomposition of measured signals. This anomaly detection approach considers the joint network and systems information of every VM gathered at the hypervisor level. Numerous researchers have successfully applied Principal Component Analysis (PCA) in their network anomaly detector methods (i.e. [49], [49], [72], [44], [48], and [10]). However, as mentioned in [64], PCA detectors are based on the assumption that the data reproduces a combined Gaussian distribution which may be unrealistic in the case of

network traffic. Researchers present in [15] a Holt-Winters forecasting method that uses the entropy of three flow features, (i.e. source IP address, destination IP address, and destination port) to model the normal traffic and exploits the Kullback-Leibler divergence distance to detect network anomalies.

3.6.2 Machine Learning Methods

Machine learning was defined in 1959 by Arthur Samuel as "the field of study that gives computers the ability to learn without being explicitly programmed" [73]. ML allows to uncovering hidden correlation patterns through an iterative learning by sample data (or past experiences) instead of being explicitly programmed. Common classes of problems that ML algorithms can solve are classification, regression, clustering, and outliers detection. Several techniques are applied for anomaly detection and based on the availability of labeled dataset an anomaly detection method can operates in Unsupervised, Semi-Supervised, or Supervised mode [14]. Supervised and semi-supervised approaches are more suitable for classification problem while unsupervised approach fits better clustering problems.

Unsupervised Anomaly Detection

Cluster analysis (or clustering) is a technique used to group objects of a similar kind into respective categories. Clustering is based on unlabeled data. In machine learning, methods that use labeled samples for the training and validation phases are said to be supervised, or semi-supervised, and methods which rely on unlabeled dataset are said to be unsupervised. Clustering can be achieved by different algorithms that vary in their notion

of what constitutes clusters and how to identify them. Usually, a clustering-based technique requires distance computation between a pair of objects. Clustering was used by [7], [59] (using K-means), and [68].

Semi-Supervised and Anomaly Detection

Both supervised and semi-supervised methods are based on knowledge provided by an external agent and they require labeled training datasets. Specifically, supervised methods need both normal and anomalous instances, and semi-supervised methods use only the normal labels. Support Vector Machines (SVM) provide a semi-supervised learning method for the anomaly detection. A flow-based anomaly detection system using a one-class SVM was proposed by [40]. The use of Multi Layer Perceptron (MLP) neural network for supervised anomaly detection systems was investigated in [41], and [81].

3.7 Flow-based anomaly detection

In literature, there has been a considerable amount of research on anomaly detection based on the flow analysis. Studies related to flow-based traffic analysis have proven useful in identifying anomalies. The authors of [49] analyze various traffic types (i.e. metrics) of sampled flows in an extensive academic network. Each traffic metric brings into focus a different set of anomalies and more specifically, this work reveal that the analysis of numbers of flows is suitable to identify *DoS*, *scan*, and *Flash Crowd*. Signal processing techniques assume that generally, traffic may be modeled as a linear state space model having Gaussian behavior. Authors in [79]

combine Kalman filtering and statistical methods for detecting volume anomalies in large-scale backbone networks. Based on the hypothesis that along their routes anomalies traverse numerous links, this approach monitors multiple links simultaneously. This technique performs anomaly detection on origin–destination (OD) flows to identify the source of the anomaly. Dewaele et al [27] extract sub-traces from randomly chosen traffic traces, model them using Gamma laws and identify the anomalous traces by tuning the deviations in the parameters of the models. Flow-based methods allow faster analysis through a reduction in data size, on the other hand, surveys have already shown how the complete absence of payload could also be the main weakness [80]. The use of these techniques makes it tough to detect attacks for which the variations are only in the payload. Nevertheless, Flow-based Intrusion Detection Systems could be used as a complement when technological constraints or privacy policy make payload-based techniques infeasible. The use of entropy estimation for anomaly detection relies on the principle that certain types of network anomalies will (meaningfully) disturb the distribution of traffic features (e.g. source/destination ports, source/destination IP addresses) [11], [9]. Authors in [11] show that packet sampling methods employ several trade-offs regarding the detection of anomalies. They analyze in detail the random packet sampling and the impact of its quantification on anomaly detection. Even if since the early 2000 packet sampling is under discussion for standardization by the IETF PSAMP working group [19], there is no standard to describe how to sample flow data. On the one hand, flow sampling decreases the computational load, on the contrary, it makes more difficult the detection process. Studies that describe the impact of flow sampling to the anomaly detection are presented by [4], [6]. Androulidakis et al. [4] utilized an entropy-based anomaly detection method based

on an intelligent flow sampling to improve the effectiveness of anomaly detection and validate it. Bartos et al. [6] proposed an adaptive flow sampling technique. An ideal sampling model used to evaluate the sampling method is provided. This approach preserves the statistics of the traffic features used for anomaly detection. In order to evaluate real anomalies in real traffic, researchers can use typically two approaches, manually checking the alert or comparing their alert events with other anomaly detectors. In the first approach, the event labeling relies on human-based knowledge, thus detected events are verified manually by the domain experts. Following this approach, authors in [9] present a histogram-based anomaly detector that identifies anomalous flows by detected the changes in traffic by applying the Kullback-Leibler divergence to several histograms that monitor distinct traffic features. Furthermore, the association rule mining allows for the extraction of the set of traffic features that describes the anomalies detected by the histograms. On the contrary, in this dissertation, we evaluate the accuracy of our anomaly detector by comparing all alert events with the results of MAWILab archive.

Authors of [38] proposed a method for anomaly detection using four flow metrics: volume of bytes; the number of packets; the number of flows to the same destination IP and port, and the number of destination ports. The validation process was performed analyzing simulated anomalies (e.g., UDP flood, ICMP flood, TCP SYN, or Port scan).

A traffic analysis focused on scan in TCP protocol was presented in [60]. The author used Nmap to perform network or port scan. The authors, in an afterward work, presented a solution to detect network anomalies (i.e. scan and flood) in a high-speed network with an accurate identification of the anomalies [61]. The key features that identify alert

consist of attack time, attacker IP and victim IP, the attack category (scan, flood, DoS, and DDoS) and the attack type (scan type, and flood type).

A two steps method to identify and classify the flows involved for the anomalies is proposed by [31]. This technique firstly determines the time slice where the anomaly occurs then find the IP flows responsible for the detected anomalous behavior.

On the contrary, instead of identifying the flows responsible of an anomaly, [24] provided for each anomaly detected, a table of all flows belonging a time slice with the highest flow frequency.

3.8 Final remarks

This chapter has provided background information on Intrusion Detection Systems and its taxonomy. Then, the Anomaly Detection Systems techniques present in the literature. Moreover, a brief overview of datasets to evaluate and validate the anomaly detection methods was provided.

Chapter 4

Big Data Analytics and Network Monitoring

4.1 Introduction

What is Big Data? According to Manyika et al. [53]definition “Big data is datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze”. This definition contains a time-variant aspect. Datasets that today can be considered as Big Data tomorrow could become “normal” data. Since this definition does not use any metric to define big data, we prefer to use Laney et al. [50] definition that defines the data growth challenge as three-dimensional, i.e., concerning an increase in Volume, Velocity, and Variety. Big Data is a relatively novel topic that has received much attention from the community in the last decade. Many research groups (such as biologists, astronomers, and computer scientists) have focused their efforts on several aspects of interest related to this

important research topic, such as methods and technologies to handle big data issues including for i.e. acquisition, storage, analytic and data mining models [3]. Among the sciences, discoveries in astrophysics have always been driven by the analysis of massive data sets. Thus, a new discipline, the Astroinformatics, has born. In this interdisciplinary field of astrophysics and computer science, scientists exploit new methods and analytics tools to solve the big data challenges faced in astronomy [30].

The number of devices connected to the network and the spread of high-speed network are leading the growth in network traffic concerning data Volume and complexity (i.e. Variety), and traffic Velocity. Network monitoring could be considered a Big Data problem as it has all the 3Vs characteristics (i.e. Volume, Velocity, and Variety). Beyond all definition of Big Data, in the context of computer networks, a Big Data problem involves a enormous amount of data (to capture, store and analyze) traversing the network infrastructure, so large that it exceed the analytic capabilities of traditional computing methods and architectures. This chapter aims at providing an overview of the new and much less investigated topic related to big data and networking. In particular, we aim at analyzing the existing literature dealing with how computer networks produce big data as well as dealing with how computer networks handle big data. For example, in the first case, network management and intrusion detection systems running distributedly across vast and high-speed networks produce a large amount of data (pertaining the status of hosts, nodes, and the network links) having the 3V properties of big data cited above. On the other hand, the several existing applications dealing with big data have more and more stringent requirements for the network and impose a huge load (in a very broad sense) on such networks.

4.2 Network Solutions for Big Data

Many architectural solutions for the various aspect of the Big Data problems are present in literature. Authors of [82] focused their work on the exponential growing of dataset sizes and the resulting data transfer problem. They proposed an efficient solution based on the Remote direct memory access over Converged Ethernet (RoCE) [5]. Most of the data transfer tools are based on TCP, with a limit of tens of Gbps on current hardware. The work presents a brief overview of the main open issue and a performance comparison between the RoCE and conventional transfer protocols (TCP, UDP), over high-speed network links. This work demonstrates that the transfers based on RoCE achieve comparable performance as TCP or UDP while maintaining a low percentage of the involved CPU resources up to two order of magnitude less than other protocols. MapReduce applications consume most than 50 percent of the bandwidth for the Shuffle, and Date-Spreading phases. To avoid potential over-subscription problems due to the uses of static routing, [86] presented a distributed adaptive routing algorithm. The authors presented an approximate Markov chain model to evaluate the convergence time of the aforementioned adaptive algorithm. This model predicts that, under the condition that the flows do not exceed the 50% of the bandwidth capacity for the edge links of the network, the algorithm would converge in a few iterations to a non-blocking routing assignment regardless the size of the network.

Due to the enormous amount of network traffic produced by batch and real-time big data processes, numbers of new architectural solutions in order to optimize the bandwidth utilization [23], [84] or reduce the network traffic [22] are proposed in the literature. One of these is Cam-

doop [22] that focuses on decreasing traffic by pushing the aggregation to the network core. In this way it reduces the amount of traffic, parallelizing the typical shuffle and reduces steps of the MapReduce processes. Camdoop is a framework for the execution of MapReduce-like processes that uses a custom transport protocol, providing a reliable communication, a specific programming for the application of the packets and the packet aggregation through the flows. It runs on CamCube [21], a platform that distributes the switch functionality between servers. Leveraging the way by which CamCube forwards the traffic, Camdoop performs data aggregation during the shuffle phase. It can be used as a Hadoop plug-in, or completely replace it since Camdoop supports the same MapReduce functions. Recent researches as [23], [84] have shown how the benefits brought by SDNs to the optimization of network utilization (e.g., using the adaptive routing) can be exploited to improve the performance of distributed applications. In [84], the author proposed a cross-layer approach for SDN controller that, by fitting the application requirements at run-time, enhances the performance of distributed application. This work combines the high-speed optical switches with SDN controllers to dynamically reconfigure them. The SDN controller functioning is based on application level information. Thus, it is interfaced to the master node (such as the Hadoop scheduler) to retrieve the job by scheduling the information used to predict the bandwidth consumption of the next application task. Then, the controller configures the underlying network routing and topology accordingly to the recognized traffic pattern, such as bulk transfers, data partitioning and aggregation (i.e. MapReduce), and low-latency control message, in order to improve the application performance at run-time. Nevertheless, flow-level traffic engineering for big data applications has been postponed by the author as future work. Among the previous work,

author of [23] proposes a network management framework that improves Big Data process through the optimization of the network utilization. This framework, named FlowComb, includes a centralized decision engine that by monitoring big data applications through software agents deployed on each application server, is able to predict the application transfer phases before they start. Then, the decision engine (as well as an SDN controller) adapts the network by changing the path using the application domain knowledge.

4.3 Big Data Solutions for network

Achieving the ability to analyze, understand and exploit information in a more efficient way is crucial for several computer network applications such as monitoring, planning, forensics, and security.

In [87], the authors propose a Big Data analytics framework for mobile network optimization (MNO). Data is collected from various sources, the users (i.e. location, mobility pattern, communication pattern, and application usage behaviors), the network core (i.e. network performance information, successful calls, application usage), and the physical level (i.e. cell information, radio signal power, and quality). The framework is composed of a BDA platform and a set of network optimization functions. These features can be useful for several tasks in MNO, such as network planning, QoE modeling, and resource allocation.

Authors in [63] propose a big data framework (i.e. Hadoop MapReduce) for change detection in temporally-evolving network traffic data. This approach involves two steps. Firstly, the traffic is sampled using a

random method then the selected time period is partitioned in temporal bins. Then, the change detection method is applied. A distributed application that uses MapReduce is implemented to perform both the sampling and the change detection phase. A performance comparison that using real traffic traces [CAIDA] shows how the distributed solution achieves better performance than a non-distributed one. However, this comparison does not take into account the accuracy or the efficiency, but only the execution time. Moreover, only a theoretical model is presented without providing a possible comprehensive implementation of the monitoring architecture. An anomaly detector referring to performance metrics of virtual machines running in a cluster is presented by [78]. They proposed a framework for real-time anomaly detection based on Apache Spark. Finally, a big data architecture for security monitoring is proposed by [54]. Data analyzed come from different sources, such as honeypots data, DNS and HTTP traffic, and IP flow records. The system proposed correlates these sources with data correlation schemes useful for network security. Finally, a performance assessment of this schemes is performed using five BDA tools, namely Hadoop, Hive, Pig, Shark, and Spark, against four scenarios (i.e. queries). This performance analysis shows how Spark achieves the best performance in all scenarios. Unfortunately, this work does not present any practical implementation for this system and the performance comparison does not consider the efficiency and the accuracy but only the computing time performance.

Among numerous others, we propose a Big Data architecture to analyze the flow network traffic in an efficient manner.

4.4 Big Data Analytics Tools

BDA frameworks, deployed on Cloud or In-House Data Center, have become critical to facing the computational demand tasks. In the following, we present an overview of the most used BDA tools in literature.

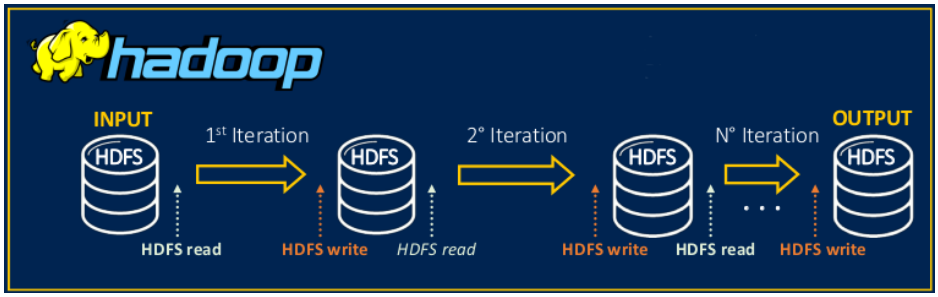

Fig. 4.1 Big Data Analytics framework - Batch Analysis

4.4.1 Apache Hadoop

Apache Hadoop is an open source distributed computing framework for distributed storage and batch processing of large data sets on clusters built from commodity hardware using simple programming models (i.e. MapReduce) [Hadoop]. It is designed to scale up from single to thousands of servers, each of which offers both local computation and storage. It allows processing big data by using batch processing. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer. Hadoop core components provide services for job scheduling (YARN), a distributed file system (HDFS), and data processing (MapReduce).

HDFS - Hadoop Distributed File System

HDFS [HDFS] is the distributed file system of Hadoop framework. HDFS is designed to store files that have the size of gigabytes to terabytes on clusters of commodity hardware. In an HDFS cluster, there are two types of nodes, namely NameNode and DataNode. The NameNode is responsible for storing and managing the metadata of files and directories in the file system. On the other hand, DataNodes, function as storage for HDFS files. In HDFS, files are split into blocks which are stored independently in a set of DataNodes. In order to provide fault tolerance and high availability, each block is replicated to multiple DataNodes. When a client application writes a file to HDFS, each block is sent to a DataNode, which then replicates it to other DataNodes. On the other hand, when a client needs a file from HDFS, each block of the file is read from the nearest one among the DataNodes hosting the block replicas [Hadoop].

4.4.2 Apache Storm

Storm is a distributed real-time computation system for processing large volumes of high-velocity data. Storm is extremely fast, with the ability to process over a million records per second per node on a cluster of modest size. Enterprises harness this speed and combine it with other data access applications in Hadoop to prevent undesirable events or to optimize positive outcomes.

4.4.3 Apache Kafka

Apache Kafka is a fast, scalable, durable, and fault-tolerant publish-subscribe messaging system. Kafka is often used in place of traditional message brokers like JMS and AMQP because of its higher throughput, reliability and replication [Kafka]. Apache Kafka can work in combination with numerous systems for real-time analysis and the rendering of streaming data, such as Apache Storm, Apache HBase or Apache Spark. Usually, it is employed for two type of application, developing real-time data workflows, exchanging messages between systems or applications in a reliable way, and real-time streaming applications that transform or react to the data stream. Kafka is a message broker horizontally scalable, and fault-tolerant. Regardless of the use case, Kafka brokers massive streams of messages for low-latency analysis in the Apache Hadoop ecosystem.

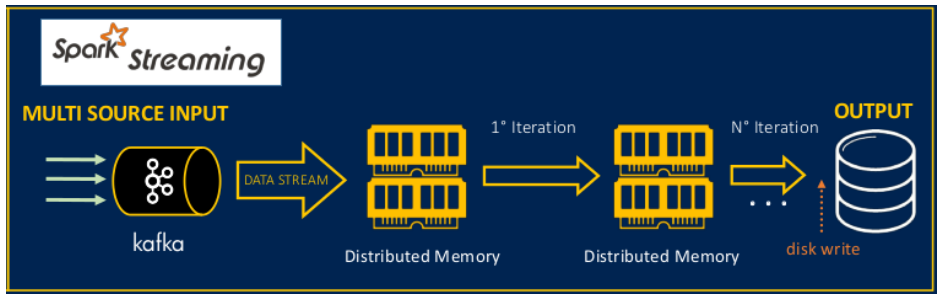

Fig. 4.2 Big Data Analytics framework - Streaming Analysis

4.4.4 Apache Spark

Apache Spark is a fast, in-memory, general purpose engine for large-scale data distributed computing. It provides development APIs to allow data

workers to efficiently execute streaming, machine learning or SQL workloads that require fast iterative access to datasets. With Spark running on Apache Hadoop YARN, developers everywhere can now create applications to exploit Spark's power, derive insights, and enrich their data science workloads within a single, shared dataset in Hadoop [85]. Moreover, it provides libraries for implementing machine learning: ML Lib, which provides machine learning models, and ML Pipelines which manages the workflow, (i.e. data preparation, post-processing, and validations) helping to develop and deploy the models.

Chapter 5

Proposed Architecture for Flow-Based Anomaly Detection

5.1 Introduction

The objective of this chapter is to give an overview of the model of the proposed architecture model. Firstly, a summary of the objectives of our system is presented in the following Section 5.2. Then, Section 5.3 gives a brief description of the architectural models (i.e. the flow monitoring architecture and the big data value chain) that inspired our multi-layer architecture. Therefore, the architecture modeling along with its implementation will be addressed in the following Section 5.2. Finally, a detailed discussion about the development of each component and their functioning are given in the sections which conclude this chapter.

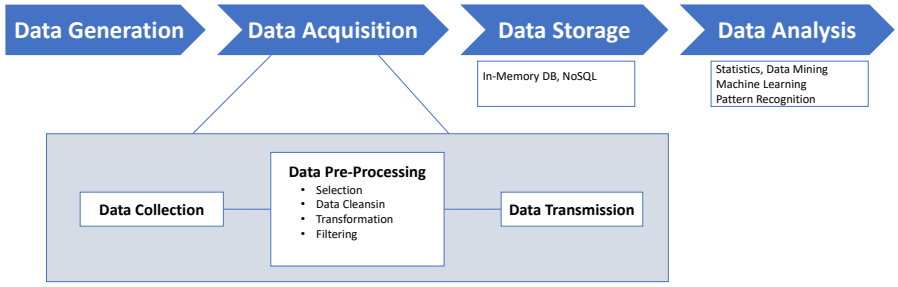

Fig. 5.1 The Big Data Value Chain

5.2 Objectives of the Proposed Architecture

The proposed architecture has been modeled with the following goals in mind:

- Capture the traffic at packet-level, to and from the monitored network, and transform it into a dataset of traffic at flow level.
- Provide an anomaly detection algorithm able to analyze the generated flow-based dataset, is able to detect the source IP responsible for the malicious behaviors.
- Reduce the response time of the anomaly detection algorithm using of Big Data Analytics framework

5.3 Proposed Architecture Model

As previously said, in this dissertation, we propose an architecture for the anomaly detection that analyzes traffic at flow level. Frameworks of batch and streaming Big Data analytics are exploited in order to enable the system to achieve better efficiency and scalability. In our proposed architecture, as in all Big Data applications, the data lifecycle can be described through the big data value chain [34]. For this reason, modeling of the architecture is loosely based on the Big Data value chain model. Figure 5.1 shows schematic workflow of the big data value chain. It transforms the raw data source into novel insights and new knowledge through different stages. The four stages constituting the Big Data value chain are Data Generation, Data Acquisition, Data Storage, and Data Analysis. A brief description of the four phases is as follows.

- Data Generation phase describes how and where data are created.
- Data Acquisition phase has mainly two aims. On the one hand, it gathers data coming from the generation phase. On the other hands, it transforms the collected data into a suitable format for the successive data storage and analysis phases. Data Acquisition is itself composed by three sub-tasks, data collection, pre-processing, and data transmission.
- Data Storage refers both to storage and management of the large dataset from the previous phase. It collects data in a convenient manner for the subsequent step of data analysis. With this purpose, data storage provides essentially two services. A hardware infrastructure responsible for warehousing of the information in a distributed, per-

sistent, and reliable manner. A data management framework that provides software interfaces to access and query the data. Moreover, the information is organized for the efficient data processing by the last stage, the Data Analysis.

- The final phase of the big data value chain is Data Analysis. It aims to extract useful insight and new knowledge and hence provides information that can aid in the decision-making process.

A high-level overview of the architecture of our systems is illustrated in Figure 5.2. In order to fulfill the objectives of the proposed model, the architecture proposed consists of two layers (i.e. Acquisition, and Big Data Analyzer). The layers are responsible for handling the phases of the Big Data value chain, namely the data acquisition, storage, and analysis.

In the first layer, the model is implemented for capture traffic packet traversing the monitored network, then transforming it in a flow records dataset. This dataset is stored in a distributed way in order to improve the subsequent analysis. In the second Layer, the model is implemented for detecting the traffic anomalies. In order to identify as soon as possible the host responsible for the anomalous traffic, a Big Data Analytics framework is employed. Th A description of the different layers that constitute the architecture will be as follows

5.3.1 Acquisition Layer

This layer is responsible for capture network traffic through the monitored network. It is used to cope te first two value chain phases (i.e. data collection, and data reduction). In this layer, the traffic is captured by a

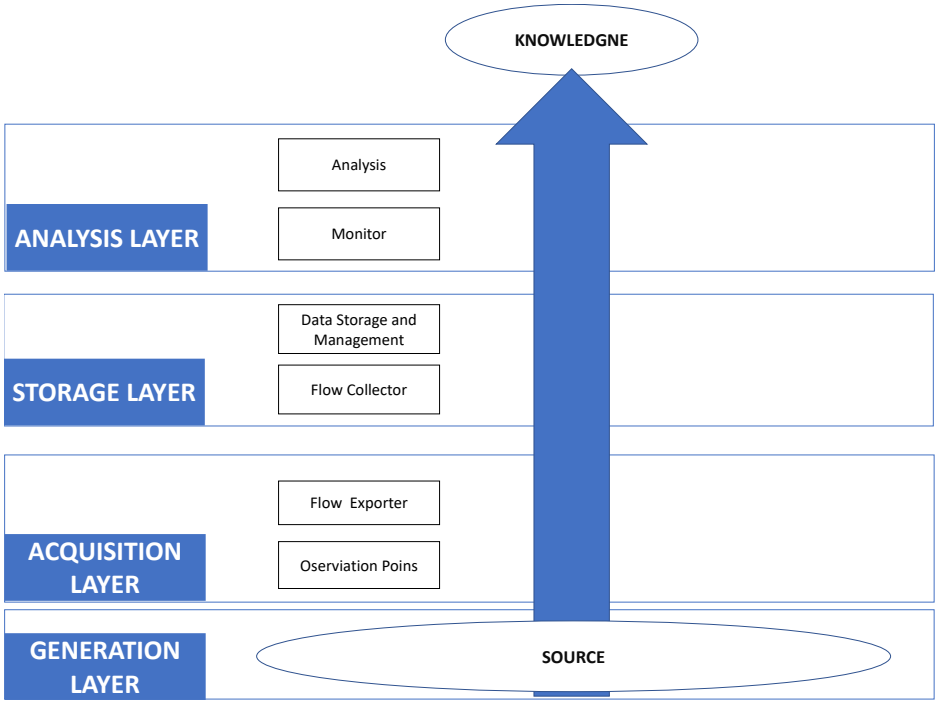

Fig. 5.2 A modeled view of the Architecture of the Proposed System

set of probes deployed in specific locations (usually at the edges of the network).Then the data is passed from the probes to the Flow Exporter that aggregate individual packets in flow records. At this point, the data can be transmitted to the next component, the Flow Collector. This layer is also responsible for the distributed storage and management of the data from the previous level and its preprocessing. With this purpose, a Flow Collector is used for gather the flow records. When it receives the flow records sent from various Flow Exporters within the network, a Flow Collector converts them into a suitable format for our detection algorithm. Finally, it stores the preprocessed data in a distributed storage system in orded to facilitate the further querying and analysis tasks.

5.3.2 Analysis Layer

This layer is composed of a big data analysis framework used to perform anomaly detection algorithms. It can receive the flow records collected by the previous Layer in batch and streaming mode.

5.4 Implementation Details

A proof of concept implementation of the proposed architecture will be provided in the following. In order to analyze the traffic at flow-level with our architecture, an appropriate setup for the Flow Exporter and the Flow Collector is needed. In our work, the role of the Flow Exporter is assigned to Softflowd [Softflowd]. Softflowd is a probe in software that is able to capture network traffic packets, and generate and export flow records. At defined time intervals, this software sends the unidirectional flow records collected to the Flow Collector. NfCapd [NFCAPD] is used as a collector node that retrieves the flow records from Softflowd. With the purpose of transforming the collected flow record into a format suitable for the analytics platform, NfDump [NFDUMP] is used. At this point, binary flow records converted into an ASCII file can be streamed directly to the analytics platform or stored in a distributed manner for further batch analysis. In both cases, the analytics platform analyzes streaming or collected data performing the anomaly detection algorithm. The algorithm receives as input dataset the list of flow records (i.e. start time, source IP address, destination IP address, number of packets, number of octets, duration, source Port, destination Port, and transport protocol). The flow records are grouped by the start time in time bins (i.e. 30 seconds). Given

a time bin, for each flow record is calculated the ratio between the number of flow generated and the number of flow received by the host identified by the source IP address. Thus, if the ratio exceed the threshold then the IP is considered as generator of anomalous flows. The detection algorithm has been deployed in the Apache Spark framework. Apache Spark allows to analyze the dataset both in batch and streaming.

The proposed architecture presented in the previous section 5.2, provides loose coupling between each stage. Thus, the collection component can be simply replaced with a different tool as well as the Flow Exporter, the probes, and the source traffic used to the analysis system.

Chapter 6

Evaluation and Results

6.1 Introduction

This chapter presents results of our proposed architecture in terms of efficiency and scalability of our method. In Section 6.2 and 6.3, the input dataset analyzed and its characterization are presented. Then, the gold standard dataset used to validate our anomaly detection method are presented in Section 6.4. The chapter concludes with the results of the evaluation of performance efficiency in Section 6.5 and scalability in Section 6.7.

6.2 The Dataset: MAWI

The MAWI (Measurement and Analysis of the WIDE Internet) traffic repository is a publicly accessible archive of real traffic from the WIDE

backbone network provided by the MAWI Working Group Traffic Archive and maintained by the WIDE Project. The MAWI archive provides anonymized packet traffic traces collected from several links. Specifically, in our evaluation, we use the packet traces captured at Samplepoint-F. Every trace collected by this link contains traffic captured every day for 15 minutes (from 14:00 to 14:15 GMT) from a transpacific backbone link connecting Japan and the United States. Different links captured the traces daily since 1999 and storing them in pcap files with anonymized IP addresses and without packets payload data. These links have been updated three times; originally the samplepoint-B was a 100 Mbps link with an 18 Mbps committed access rate (CAR). It was replaced in July 2006 by a full 100 Mbps link (i.e. the samplepoint-F) that was updated to a 1Gbps link with a 150 Mbps CAR in June 2007. Finally, in June 2016 the CAR was officially removed. Summarizing the contributions of MAWI dataset for Internet traffic research are threefold:

- Accessible - the entire archive is freely available to all researchers. Therefore, they are able to replicate and compare results achieved employing MAWI traces as input dataset.
- Extensive - MAWI allows investigating about the evolution of Internet traffic since its daily monitoring lasting more than 15 years which corresponds to an archive of tens of Terabyte (TB) of backbone network traffic traces.
- Various - during the 15 minutes a daily trace, MAWI monitors hundreds of thousands IP addresses that use several applications. Moreover, the archive includes real traffic since 2001 to nowadays.

Thus, every trace carries within several kinds of anomalies, from well-known attacks to unknown and new threats.

The analyzed flow dataset was generated using the MAWI traffic traces captured from the samplepoint-F between October the 1st and October 31th, 2014. These packet traces were transformed in ASCII datasets using the aforementioned flow exporting functionality of our architecture.

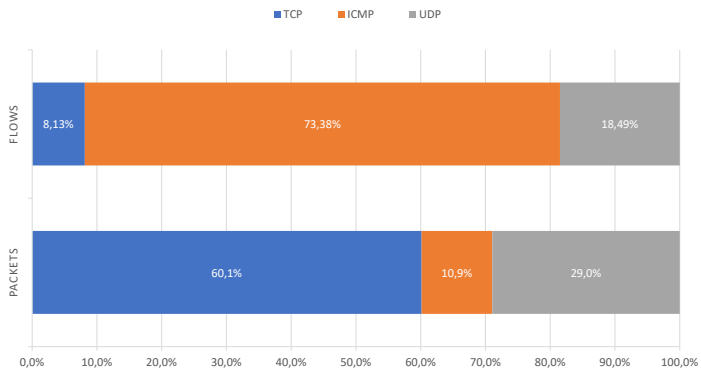

Fig. 6.1 The MAWI dataset breakdown according to the protocol at packet and flow level

Figure 6.1 shows the breakdown of input dataset according to the level 3 protocols at packet and flow level, respectively. The collected dataset presents a subdivision in only three IP protocols: ICMP, TCP, and UDP. At packet level, the majority of traffic have TCP protocol (i.e. almost 68% of the packets). A second slice of the dataset consists of UDP traffic (i.e. almost 30% of the packets). Finally, the lower fraction is due to ICMP (i.e. 10% of packet traffic). The packet level protocol breakdown is in contradiction with protocol breakdown at flow level. In this case, the protocol distribution is the opposite of the previous one.

The majority of traffic is due to ICMP with the 73% of flows, then UDP protocol constitutes about the 20% of flow traffic, finally only the 8% of flows have TCP protocol.

6.3 Dataset characterization

In order to characterize the MAWI traffic traces, in the following, we analyze and study the dataset in various flow dimensions, namely production rate, duration, the number of packets, flow size, and packet rate. Specifically, production Rate refers to the number of flows per minute produced by every source IP. Duration is the time elapsed between the first and the last packet belonging the flow. The flow size refers to the volume of bytes transferred with a single flow.

We analyze IP addresses where the production rate exceeds the threshold of ten thousand flows per minute. Thus, the dataset refers to a list of 10 source hosts IP addresses. The feature on which to apply the threshold (i.e. production rate) and the value of the threshold have been chosen in an empirical way. Thus, as can be seen in Figure 6.2, using a higher threshold removes the only source IPs that appear for more than one slice of time (i.e. 202.11.241.164 and 220.48.217.28). On the other hand, using a threshold lower than 2000 flow per minute, the number of sources that will be involved can exceed the hundreds of hosts. In the following, we discuss traffic flows characterization for these set of source IPs.

6.3.1 Flow per Minute

Figure 6.2 shows for each source host, identified by their IP addresses, the number of flows per minute generated. Most of these hosts, (i.e. eight out of ten) have a transient characterization. The faster one generates about one hundred of thousand of flows in less than 30 seconds.

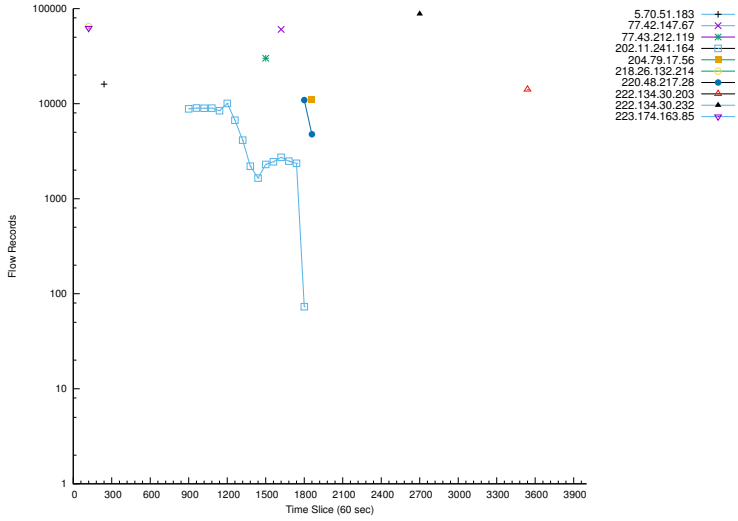

Fig. 6.2 Number of flows generated by the top 10 source IP with respect to time slices.

6.3.2 Duration

Figure 6.3 shows for each source IP the average duration of the flows generated in each time slice. Most of them are characterized by a duration practically equal to 0 seconds.

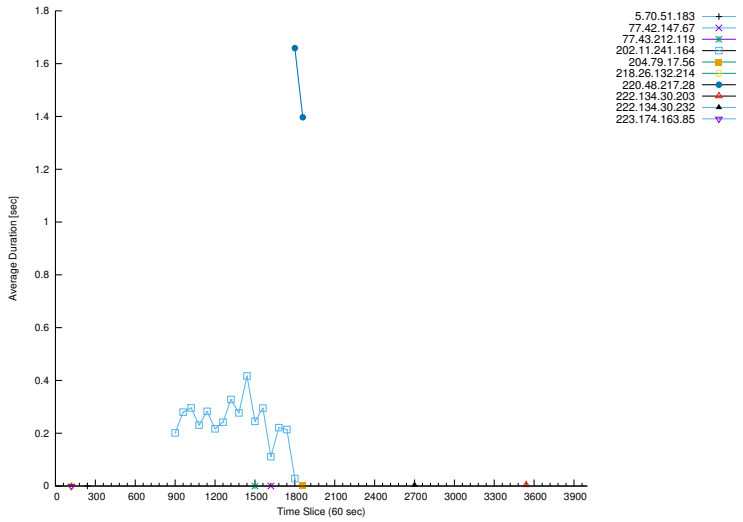

Fig. 6.3 Average duration of flows generated by the top 10 source IPs with respect to time slice.

6.3.3 Number of Ports

Figures 6.4 and 6.5 show respectively the number of source ports and the number of destination port used by the hosts inside the monitored network at each slice of time. The behavior of hosts with source IP addresses 220.48.217.28, and 222.134.30.232 deserves more in-depth analysis. The first uses hundreds of source ports but only one destination port (i.e. 2967). The second, with the source IP with the higher number of flows across the entire trace, uses only ports 1026 and 1027.

6.3.4 Bytes and Packets Transferred

Figure 6.6 and 6.7 depicted the number of bytes and the number of packets transferred by each IP in every time slice (one minute), respectively.

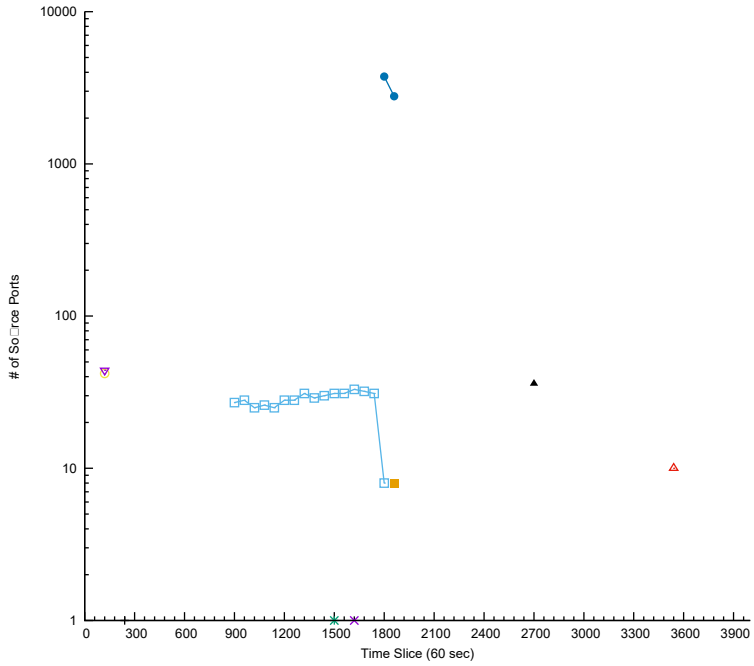

Fig. 6.4 Number of source port involved by the top 10 source IP with respect to time slices

6.4 The Gold Standard: MAWILab

The MAWILab project provides an archive of labeled anomalies in the samplepoint-F traffic traces of the MAWI archive. To labels the anomalous events in the samplepoint-F traces, MAWILab used an advanced unsupervised graph-based methodology that compares and combines four different and independent anomaly detectors to provide the labeled dataset. The used detectors are respectively based on the Hough transform [29], the Gamma distribution [27], the Kullback-Leibler divergence [9], and the Principal Component Analysis (PCA) [45]. According to the analysis performed the traffic is labeled as following:

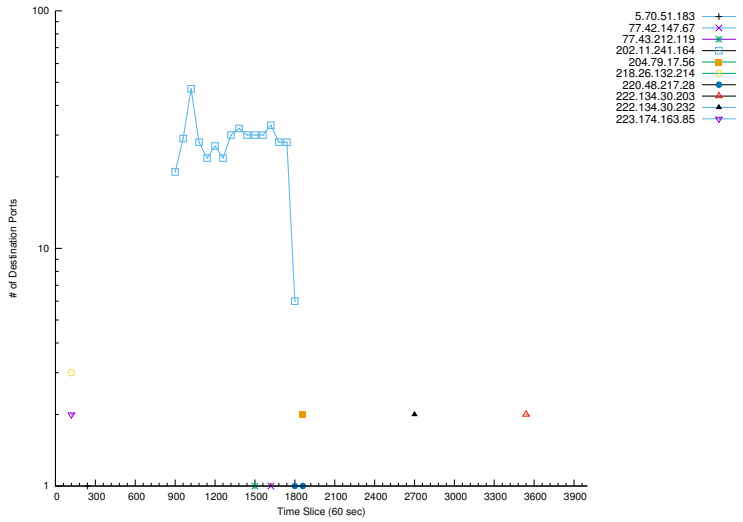

Fig. 6.5 Number of destination port involved by the top 10 source IP with respect to time slices.

- Anomalous if traffic is certainly identified as abnormal;
- Suspicious if traffic is not clearly identified as anomalous.
- Notice if at least one of the four anomaly detectors reported the traffic as abnormal;
- Benign for normal traffic that is not detected by any of the detectors.

MAWILab archive is daily updated to include the new traffic traces upcoming from the MAWI Archive [MAWI]. Due to probabilistic nature of the MAWILab classification method, we are not sure that all the anomalies in the traffic traces are actually detected and so presents in the MAWILab dataset. However, this database is one of the most used gold standard [83] for researchers to evaluate their traffic anomaly detection methods.

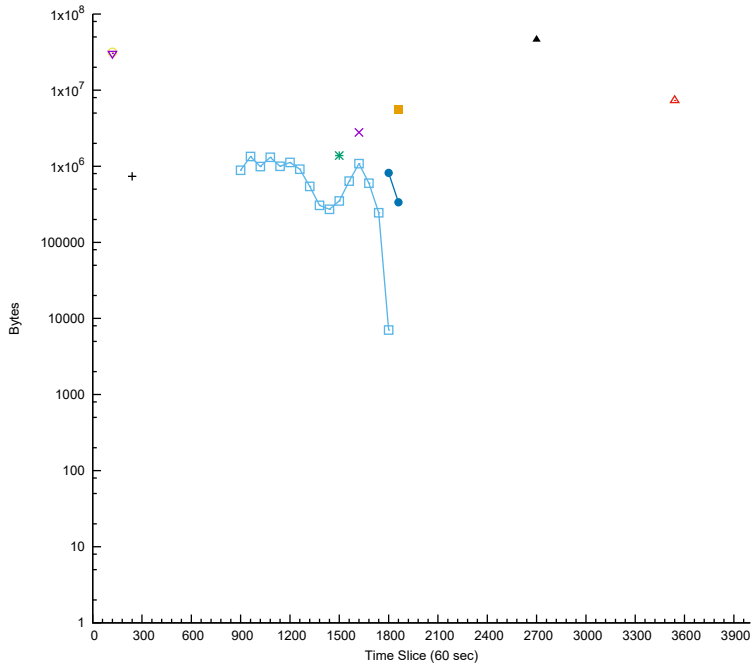

Fig. 6.6 Number of bytes sent by the top 10 source IPs with respect to time slices.

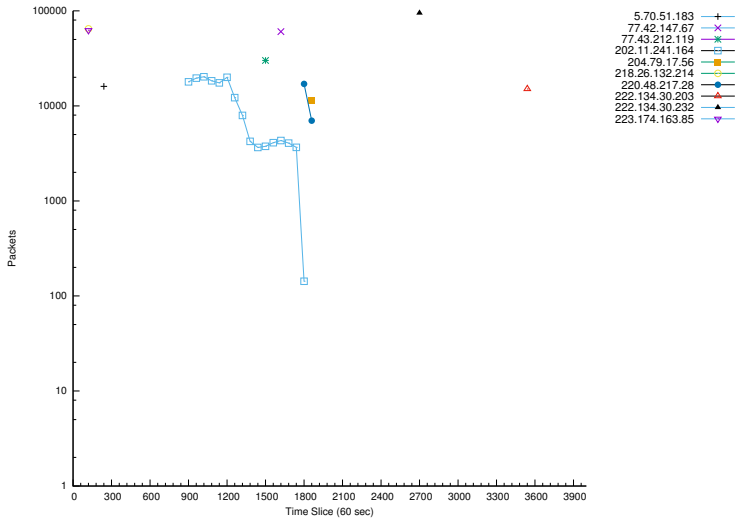

Fig. 6.7 Number of packets sent by the top 10 source IPs with respect to time slices.

For each trace are available two CSV files. One file containing the traffic labeled as anomalous or suspicious, the other one identify the traffic labeled as notice. Each row of these files consists of a 4-tuple (i.e. source IP address, source Port, destination IP address, destination Port) describing the traffic characteristics and additional information such as the heuristic and taxonomy classification results. The CSV file header gives the actual order of the aforementioned information [29]:

- **anomalyID** a unique anomaly identifier. Several lines in the CSV file can describe different sets of packets that belong to the same anomaly. The anomalyID field permits to identify lines that refer to the same anomaly.
- **srcIP** is the source IP address of the identified anomalous traffic (optional).

- **srcPort** is the source port of the identified anomalous traffic (optional).
- **dstIP** is the destination IP address of the identified anomalous traffic (optional).
- **dstPort** is the destination port of the identified anomalous traffic (optional).
- **taxonomy** is the category assigned to the anomaly using the taxonomy for backbone traffic anomalies.
- **heuristic** is the code assigned to the anomaly using a simple heuristic based on port number, TCP flags, and ICMP code.
- **distance** is the difference Dn-Da.
- **nbDetectors** is the number of configurations (detector and parameter tuning) that reported the anomaly.
- **label** is the MAWILab label assigned to the anomaly, and it can be either anomalous, suspicious, or notice.

6.4.1 Taxonomy

The information stored in the CSV files mentioned above concerning the taxonomy of the anomalies are out by a process described in [57]. The authors define this taxonomy in order to provide a more detailed classification of events detected in MAWILab repository. The taxonomy creation process is initiated by exploiting the knowledge of network anomalies, then continues iteratively refining from time to time the description of

detected events but not yet classified by taxonomy. Figure 6.8 shows a high view of the taxonomy structure. The taxonomy elements are coarsely divided into two categories: normal, and anomalous event. Normal events are further divided into heavy hitter, point to multipoint behaviors and other events (such as tunneling, and outages), while anomalous events include denial of service, whether distributed ones or not, and port and network scans event. Each event can match one or more labels. In this case, it is chosen as label the one furthest from the root of the taxonomy tree.

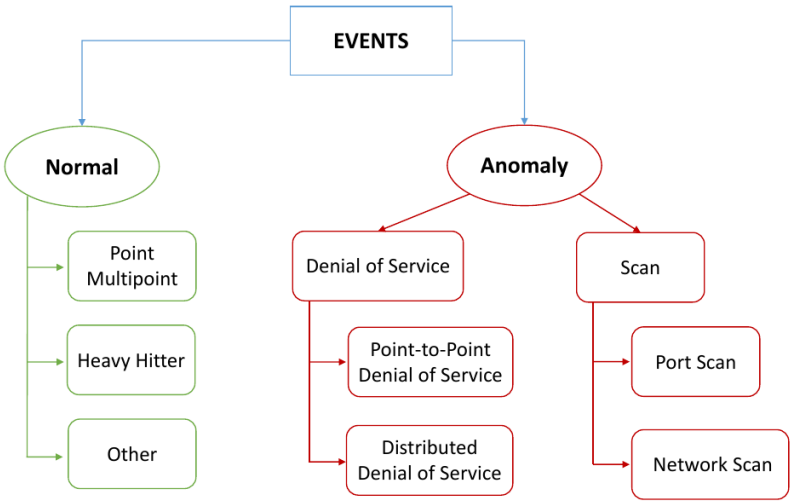

Fig. 6.8 The structure of MAWILab for the anomaly classification taxonomy

6.4.2 MAWILab Characterization

Starting from the taxonomy defined by [57], we have taken into account only the anomalies that are related to malicious behaviors. All this anoma-

Table 6.1 MAWILab - Attacks Percentage

Taxonomy	**2007**	**2008**	**2009**	**2010**	**2011**	**2012**	**2013**	**2014**	**2015**	**2016**
PortScan	0.7%	10%	1.6%	3%	3.6%	4.2%	2.3%	2.3%	1.3%	0.8%
ICMPNetScan	4.9%	2.5%	3.1%	2.1%	1.1%	0.7%	0.6%	0.1%	0.6%	0.8%
TCPNetScan	56%	50%	64%	61%	58%	63%	65%	74%	72%	74%
UDPNetScan	35%	30.1%	30%	30%	25%	26.8%	28%	21%	23%	32%
DoS	2%	1.3%	1.32%	2.4%	11%	4.7%	1.6%	2.4%	2.2%	1.8%

lies belonging to the following classes: DoS/DDoS, Port Scan, and Network Scan. Afterward, we filtering out from the dataset all the anomalies that involving ICMP protocol. Thus, we have identified four anomalies categories. Figure 6.9 shows the percentages of these anomaly classes founded by MAWILab from 2007 to 2016. Figure 6.9 depicts that only a small part of anomalies (i.e. about the 3% of anomalies) involved ICMP traffic. On the other hand, as showed in Table 6.2, the MAWI traces are composed on average of 80% ICMP flow traffic. For this reason, we filter out the ICMP traffic from all traces.

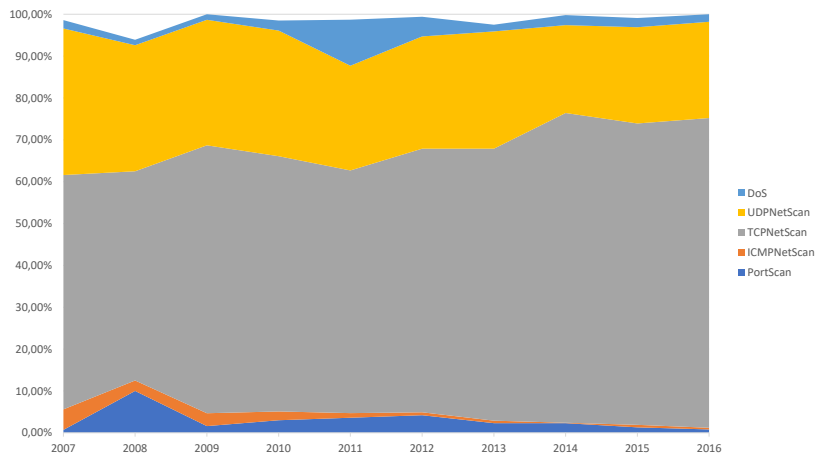

Fig. 6.9 Malicious anomalies percentage in MAWILab archive

Table 6.2 MAWI Protocol Breakdown

Protocol	Packets (MAWI)	Flows (MAWI)	Flows no-ICMP
TCP	60.13%	8.13%	30.54%
ICMP	10.92%	73.38%	-
UDP	28.95%	18.49%	69.46%

6.5 Efficiency Evaluation

In this section the proposed anomaly detection method is evaluated using alarms reported by the MAWILab archive. In particular, the positive precision rate of the detector is discussed.

6.5.1 Evaluation Metrics

In order to evaluate the efficiency of an anomaly detection method can be used various metrics, namely Accuracy, Precision, Recall (or True Positive Rate), and False Alarm Rate. These four metrics are defined as follows.

Accuracy $\frac{t_p+t_n}{t_p+t_n+f_p+f_n}$

Precision $\frac{t_p}{t_p+f_p}$

Recall or True Positive Rate $\frac{t_p}{t_p+f_n}$

False Alarm Rate $FPR = \frac{f_p}{N} = \frac{f_p}{t_n+f_p}$

Accuracy is not appropriate for evaluating methods for rare event detection. For example, if the Network traffic dataset has the 99.99% of normal data and 0.01% of intrusions, a trivial classifier (where everything is labeled as normal) can achieve 99.99% accuracy!

6.6 Experimental Testbed

To evaluate the effectiveness of our proposed architecture and to validate the anomaly detection algorithm we developed a testbed. In this testbed, flow exporters, and flow collector are simulated through two open source software, respectively Softflowd and Nfcapd. Using Softflowd as Flow Exporter, real traffic traces are transformed into flow records and exported to the flow collector (i.e. Nfcapd). Then, using Nfdump the collected flow traces are transformed into a suitable file format for the Big Data Analytics framework. These files can be stored using Hadoop Distributed File System (HDFS) for batch analysis or streamed to the BDA framework through a message broker (i.e. Apache Kafka) for streaming analysis. Finally, Apache Spark, could analyzes streaming or collected data, running our anomaly detection algorithm.

6.6.1 Experimental Result

To validate our algorithm we used all traces of October 2014 from MAWI archive. System performance has been calculated, for each traffic trace analyzed, comparing source IP addresses (i.e hosts source of anomaly flows) identified by our detection algorithm with the result of MAWILab archive. MAWILab archive is used as a gold standard to validate our

method. Table 6.3 depicts the confusion matrix with the positive precision rate achieved by our algorithm comparing its results with MAWILab. The results showed a high false positive rate. Due to probabilistic nature of MAWILab classification methods, we are not sure that all attacks in the traffic trace are actually in MAWILab the dataset. For this reason, we performed for all False Positive IP a rule-based refinement process. The rule-based process aims to detect the port scan and network scan activities. Thus, the second part of the Table 6.3 shows the confusion matrix when the refinement process is applied. The results in Figure 6.10 shows that our algorithm achieving an average precision of 90%.

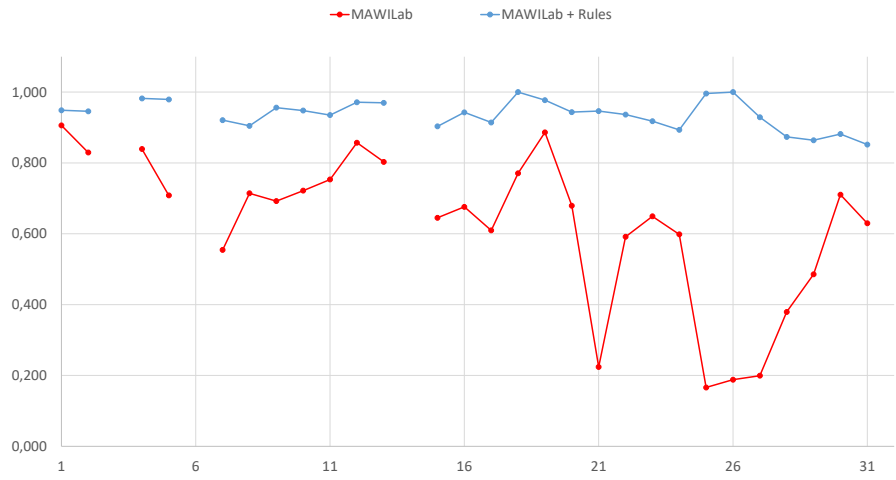

Fig. 6.10 Algorithm Precision Comparison

Table 6.3 Confusion Matrix - MAWILab as gold standard vs MAWILab plus refinement rule-based process

	MAWILab			**Refinement**			**PercentageChange**
Day	**TP**	**FP**	**PPV**	**TP**	**FP**	**PPV**	
1	106	11	0.906	111	6	0.949	4.7%
2	107	22	0.829	122	7	0.946	14.0%
4	47	9	0.839	55	1	0.982	17.0%
5	68	28	0.708	94	2	0.979	38.2%
7	56	45	0.554	93	8	0.921	66.1%
8	60	24	0.714	76	8	0.905	26.7%
9	63	28	0.692	87	4	0.956	38.1%
10	83	32	0.722	109	6	0.948	31.3%
11	58	19	0.753	72	5	0.935	24.1%
12	30	5	0.857	34	1	0.971	13.3%
13	53	13	0.803	64	2	0.970	20.8%
15	60	33	0.645	84	9	0.903	40.0%
16	73	35	0.676	99	6	0.943	39.5%
17	64	41	0.610	96	9	0.914	50.0%
18	37	11	0.771	48	0	1	29.7%
19	78	10	0.886	86	2	0.977	10.3%
20	72	34	0.679	100	6	0.943	38.9%
21	71	246	0.224	300	17	0.946	322.5%
22	84	58	0.592	133	9	0.937	58.3%
23	87	47	0.649	123	11	0.918	41.4%
24	73	49	0.598	109	13	0.893	49.3%
25	41	206	0.166	246	1	0.996	500.0%
26	47	203	0.188	250	0	1	431.9%
27	59	237	0.199	275	21	0.929	366.1%
28	33	54	0.379	76	11	0.874	130.3%
29	50	53	0.485	89	14	0.864	78.0%
30	54	22	0.711	67	9	0.882	24.1%
31	34	20	0.630	46	8	0.852	35.3%

6.6.2 Rule-based Refinement

For each IP in the list of the false positives, either source or destination, it is performed a *refinement* rule-based process. This operation consists of analyzing other network traffic descriptors to detect abnormal patterns that none of the MAWI anomaly detectors was able to identify. The traffic descriptors manually analyzed are the following:

- the rate between the number of generated flows and the number of destination ports contacted.
- the rate between the number of generated flows and the number of destination IP addresses contacted.

Therefore, IPs that present the following patterns have been labeled as "possible" anomalies:

- the ratio between the number of generated flows and the number of destination ports connected is less than two we have a port scan
- the ratio between the number of generator flows and the destination IP addresses contacted is lower than two, then we have a Network Scan
- many source IPs labeled as anomaly contact the same set of destination IP, then we have a DDoS

Table 6.4 Cloud Cluster Configurations

Configuration	# of Nodes	RAM Memory	Total Memory
c1 = original	2	8 GB (m4.large)	16 GB
c2 = scale out	4	8 GB (m4.large)	32 GB
c3 = scale up	2	16 GB (m4.xlarge)	32 GB

6.7 Performance Evaluation

Since our architecture aims to manage big data traffic, a performance evaluation in terms of response time is presented.

6.7.1 Testbed Setup

We prepared three cluster configurations of Amazon Amazon Elastic Map Reduce which are used to define the nodes of our distributed anomaly detection architecture. Each cluster configuration is reported in Table 6.4.

For each configuration, the response time is evaluated processing four traffic traces of (i.e. t1 = 15, t2 = 30, t3 = 60, and t4 = 120 minutes of flow traffic). Figure 6.11 shows the execution time of each cluster configuration. The difference are noticeable with the increasing of the size of the traces analyzed with configuration C3 (i.e. scale up) that shows the better performance.

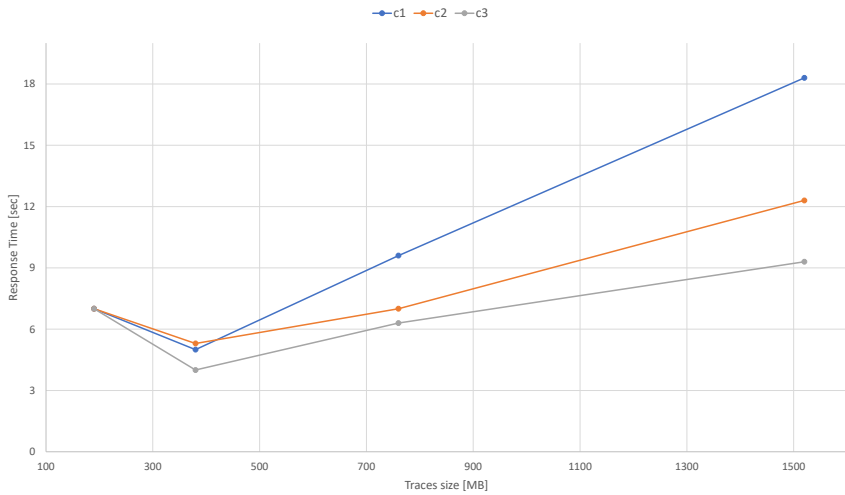

Fig. 6.11 Comparison of response time among the cluster configurations

6.8 Final Remarks

Statistical methods have been proposed for network traffic anomaly detection due to the capacity of real-time detecting of both known and unknown attacks without attack signatures. In this chapter, we proposed the rate between flows generated and flows received by each host monitored as a new feature for traffic anomaly detection and introduced an unsupervised statistical anomaly detection method based on this new proposed feature.

We tested the efficiency of the proposed feature and method with real-world backbone traffic traces containing several kinds of volumetric anomalies (such as Network Scans, Port Scans, and DoS) captured by the MAWI Working Group Traffic Archive.

The results showed that the proposed method could detect more anomalies than the four MAWILab methods, namely Hough transform, the

Gamma distribution, the Kullback-Leibler divergence, and the Principal Component Analysis, in terms of false positive rates and precision, with an average positive predictive value of about 90%.

Finally, an evaluation of the scalability performance of our architecture was presented. The results showed that the proposed architecture deployed in a private cloud given better performance in scale up than in scale out.

Chapter 7

Conclusion

7.1 Review of findings and contributions

In this thesis, we faced the challenge of detecting network anomalies in huge traffic datasets. We have introduced, designed, and developed an innovative architecture for anomaly detection in high-speed networks. The processing and storage resources to monitor high-speed network can be very expensive due to the data volume. To cope with this issue, our system analyzes the network traffic at flow-level instead of the single packet information. We focused on detecting the hosts responsible for the anomalous flows, instead of detecting collective anomalies as most of the literature [4, 49, 13, 42, 60]. In Chapter 6 we described the implementation of the proposed architecture. We developed an experimental testbed which by exploiting Apache Spark eased the computational scalability problem. The proposed detector algorithm has been validated by detecting real anomalies in real backbone traffic traces captured by the MAWI group.

For each input trace, we have compared our results with the ones provided by the MAWILab anomalies dataset. Due to the probabilistic nature of the classification methods used by MAWILab, we considered their results as a gold standard rather than a ground truth. With this purpose, we have performed a refinement process for false positive data. In this phase we analyze our false positives with a rule-based detection method to figure out which of these events are actually false alarms and which anomalies. The rules applied are designed to identify network and port scan attacks. The results of this process has shown that our method achieve an average precision of 90% in detecting volumetric anomalies, such as Net Scan, Port Scan, and DoS/DDoS, and a lower false negative rate compared to the four anomaly detectors used by MAWILab. Finally, we have investigated on the detection time as function of the data volume variation. We developed a cloud version of our testbed using Amazon Elastic Map Reduce employing three cluster configuration in order to evaluate the architecture scaling out and up performance. The results have shown that the better performance are achieved when our architecture was made to scale vertically (i.e. when more memory resources are added). In the summary, the key contributions presented in this thesis are as the following:

- It proposes a novel methodology for network anomaly detection in high-speed networks, witch analyzing flow traffic. This methodology does not require a prior knowledge, such as a training dataset or a database of attacks behaviors.

- In order to validate the methodology, it provides a long term analysis on real backbone traces.

- It provides an architecture that exploiting a Big Data Analytics framework is able to improve the processing performance. Thus, the algorithm mentioned above is able to detect anomalous events in real backbone traces with a response time real-time like.

7.2 Future Works

Anomaly detection is a very vast subject, and in this dissertation, we have only covered some of the many challenges that network operators have to face. The architecture in Chapter 5 exclusively deal with traces from the samplepoint F of the MAWI archive, where all the traces belonging to this dataset only lasts for 15 minutes. These come with the disadvantage that it is impossible to identify events that have a longer duration. As in the Chapter 6, the main advantage of our method is the easy identification of the host responsible for anomalous event. But, no taxonomy nor information regard to the type of anomaly is provided. However, the use of our anomaly detection method in combination with a system for the taxonomy definition might be used to facilitate the interpretation of events by network administrators.

References

[Abilene] Abilene. Abilene Network. http://cs-www.bu.edu/fac/crovella/abilene-distro.tar. Last Accessed: 2017-02-17.

[2] Abt, S. and Baier, H. (2014). Are we missing labels? a study of the availability of ground-truth in network security research. In *Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), 2014 Third International Workshop on*, pages 40–55. IEEE.

[3] Amaro, V., Angora, G., Brescia, M., Cavuoti, S., De Cicco, D., Delli Veneri, M., Garofalo, M., Longo, G., Nocella, A., Riccio, G., and Vellucci, C. (2017). Big Data in multidisciplinary scientific interoperable data mining framework. In *Astroinformatics 2016 - Proceedings IAU Symposium*, number 325.

[4] Androulidakis, G., Chatzigiannakis, V., and Papavassiliou, S. (2009). Network anomaly detection and classification via opportunistic sampling. *IEEE Network*, 23(1):6–12.

[5] Association, I. T. et al. (2010). Infiniband architecture specification release 1.2. 1 annex a16: Roce. *InfiniBand Trade Association*.

[6] Bartos, K., Rehak, M., and Krmicek, V. (2011). Optimizing flow sampling for network anomaly detection. In *2011 7th International Wireless Communications and Mobile Computing Conference*, pages 1304–1309. IEEE.

[7] Bernaille, L., Teixeira, R., Akodjenou, I., Soule, A., and Salamatian, K. (2006). Traffic classification on the fly. *ACM SIGCOMM Comput. Commun. Rev.*, 36(2):23–26.

[8] Botta, A., Avallone, A., Garofalo, M., and Ventre, G. (to be published). Internet streaming and network neutrality: comparing the performance of video hosting services. *Submitted to Computer Communications*.

[9] Brauckhoff, D., Dimitropoulos, X., Wagner, A., and Salamatian, K. (2012). Anomaly Extraction in Backbone Networks Using Association Rules. *IEEE/ACM Trans. Netw.*, 20(6):1788–1799.

[10] Brauckhoff, D., Salamatian, K., and May, M. (2009). Applying PCA for traffic anomaly detection: Problems and solutions. *Proc. - IEEE INFOCOM*, pages 2866–2870.

[11] Brauckhoff, D., Salamatian, K., and May, M. (2010). A Signal Processing View on Packet Sampling and Anomaly Detection. *Proc. - IEEE INFOCOM*.

[CAIDA] CAIDA. Center for Applied Internet Data Analysis (CAIDA) Dataset. http://www.caida.org/data/overview. Last Accessed: 2017-02-17.

[13] Casas, P., Vaton, S., Fillatre, L., and Nikiforov, I. (2010). Optimal volume anomaly detection and isolation in large-scale IP networks using coarse-grained measurements. *Comput. Networks*, 54(11):1750–1766.

[14] Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Comput. Surv.*, 41(3):15:1–15:58.

[15] Chang, S., Qiu, X., Gao, Z., Qi, F., and Liu, K. (2010). A flow-based anomaly detection method using entropy and multiple traffic features. In *Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on*, pages 223–227. IEEE.

[16] Cho, K., Mitsuya, K., and Kato, A. (2000). Traffic Data Repository at the WIDE Project. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '00, pages 51–51, Berkeley, CA, USA. USENIX Association.

[17] Cisco (2016). Cisco VNI Forecast and Methodology, 2015-2020. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html. Last Accessed: 2017-02-17.

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

[19] Claise, B., Johnson, A., and Quittek, J. (2009). Packet Sampling (PSAMP) Protocol Specifications. RFC 5476, RFC Editor.

[20] Claise, B., Trammell, B., and Aitken, P. (2013). Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC 7011, RFC Editor.

[21] Costa, P., Donnelly, A., and Rowstron, A. (2010). CamCube: a key-based data center. *Microsoft Research*.

[22] Costa, P., Donnelly, A., Rowstron, A., and Shea, G. O. (2012). Camdoop : Exploiting In-network Aggregation for Big Data Applications. *NSDI'12 Proc. 9th USENIX Conf. Networked Syst. Des. Implement.*, pages 1–14.

[23] Das, A., Lumezanu, C., Zhang, Y., Singh, V., Jiang, G., and Yu, C. (2013). Transparent and Flexible Network Management for Big Data Processing in the Cloud. In *Present. as part 5th USENIX Work. Hot Top. Cloud Comput.* USENIX.

[24] De Assis, M. V. O., Rodrigues, J. J. P. C., and Proenca, M. L. (2013). A novel Anomaly Detection system based on seven-dimensional flow analysis. In *GLOBECOM - IEEE Glob. Telecommun. Conf.*, pages 735–740, Atlanta, GA. IEEE.

[25] Debar, H., Dacier, M., and Wespi, A. (1999). Towards a taxonomy of intrusion-detection systems. *Comput. Networks*, 31(8):805–822.

[26] Denning, D. (1987). An Intrusion-Detection Model. *IEEE Trans. Softw. Eng.*, SE-13(2):222–232.

[27] Dewaele, G., Fukuda, K., Borgnat, P., Abry, P., and Cho, K. (2007). Extracting Hidden Anomalies using Sketch and Non Gaussian Multiresolution Statistical Detection Procedures. In *Proc. ACM SIGCOMM Work. Large-Scale Attack Def.*, pages 1–8, New York, New York, USA. ACM Press.

[28] Fang, H., Zhang, Z., Wang, C. J., Daneshmand, M., Wang, C., and Wang, H. (2015). A survey of big data research. *IEEE Netw.*, 29(5):6–9.

[29] Fontugne, R., Borgnat, P., Abry, P., and Fukuda, K. (2010). MAW-ILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. In *Proceedings of the 6th International Conference*, Co-NEXT '10, pages 8:1–8:12, New York, NY, USA. ACM.

[30] Garofalo, M., Botta, A., and Ventre, G. (2017). Astrophysics in the Big Data Era: Challenges, Methods, and Tools. In *Astroinformatics 2016 - Proceedings IAU Symposiu*, number 325.

[31] Glatz, E. and Dimitropoulos, X. (2012). Classifying internet one-way traffic. *ACM SIGMETRICS Performance Evaluation Review*, 40(1):417.

[32] Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660.

[Hadoop] Hadoop. The Apache Hadoop Project. http://hadoop.apache.org. Last Accessed: 2017-02-17.

[34] Han Hu, Yonggang Wen, Tat-Seng Chua, and Xuelong Li (2014). Toward Scalable Systems for Big Data Analytics: A Technology Tutorial. *IEEE Access*, 2:652–687.

[HDFS] HDFS. Apache Hadoop Distributed File System. http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html. Last Accessed: 2017-02-17.

[36] Hodge, V. and Austin, J. (2004). A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, 22(2):85–126.

[37] Hoque, N., Bhuyan, M. H., Baishya, R. C., Bhattacharyya, D. K., and Kalita, J. K. (2014). Network attacks: Taxonomy, tools and systems. *J. Netw. Comput. Appl.*, 40(1):307–324.

[38] Huy Anh Nguyen, Tam Van Nguyen, Dong Il Kim, and Deokjai Choi (2008). Network traffic anomalies detection and identification with flow monitoring. In *2008 5th IFIP International Conference on Wireless and Optical Communications Networks (WOCN '08)*, pages 1–5, Surabaya. IEEE.

[39] IDC (2016). Smartphone OS Market Share, 2016 Q3. http://www.idc.com/promo/smartphone-market-share.

[40] Ippoliti, D. and Zhou, X. (2014). Online Adaptive Anomaly Detection for Augmented Network Flows. In *2014 IEEE 22nd Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst.*, pages 433–442. IEEE.

[41] Jadidi, Z., Muthukkumarasamy, V., Sithirasenan, E., and Sheikhan, M. (2013). Flow-Based Anomaly Detection Using Neural Network Optimized with GSA Algorithm. In *2013 IEEE 33rd Int. Conf. Distrib. Comput. Syst. Work.*, pages 76–81, Philadelphia, PA. IEEE.

[42] Johnson, T. and Lazos, L. (2014). Network anomaly detection using autonomous system flow aggregates. In *2014 IEEE Glob. Commun. Conf.*, pages 544–550. IEEE.

[Kafka] Kafka, A. A high-throughput, distributed messaging system. http://kafka.apache.org. Last Accessed: 2017-02-17.

[44] Kanda, Y., Fontugne, R., Fukuda, K., and Sugawara, T. (2013). ADMIRE: Anomaly detection method using entropy-based PCA with three-step sketches. *Comput. Commun.*, 36(5):575–588.

[45] Kanda, Y., Fukuda, K., and Sugawara, T. (2010). Evaluation of anomaly detection based on sketch and PCA. *GLOBECOM - IEEE Glob. Telecommun. Conf.*, pages 1–5.

[46] Karagiannis, V., Chatzimisios, P., Vazquez-Gallego, F., and Alonso-Zarate, J. (2015). A survey on application layer protocols for the internet of things. *Transaction on IoT and Cloud Computing*, 3(1):11–17.

[KDD] KDD. The third international knowledge discovery and data mining tools competition dataset (KDD Cup 1999 data).

[48] Kudo, T., Morita, T., Matsuda, T., and Takine, T. (2013). PCA-based robust anomaly detection using periodic traffic behavior. In *2013 IEEE Int. Conf. Commun. Work.*, pages 1330–1334. IEEE.

[49] Lakhina, A., Crovella, M., and Diot, C. (2004). Characterization of network-wide anomalies in traffic flows. In *Proc. 4th ACM SIGCOMM Conf. Internet Meas. - IMC '04*, number 6, page 201, New York, New York, USA. ACM Press.

[50] Laney, D. (2001). 3-d data management: controlling data volume, velocity and variety. *Appl. Deliv. Strateg.*, 949(February 2001):4.

[51] Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., and Das, K. (2000). The 1999 DARPA off-line intrusion detection evaluation. *Comput. Networks*, 34(4):579–595.

[52] Malkin, G. (1993). Traceroute Using an IP Option. RFC 1393, RFC Editor.

[53] Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., and Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity.

[54] Marchal, S., Jiang, X., State, R., and Engel, T. (2014). A Big Data Architecture for Large Scale Security Monitoring. In *2014 IEEE International Congress on Big Data*, pages 56–63. IEEE.

[55] Marnerides, A. K., Spachos, P., Chatzimisios, P., and Mauthe, A. U. (2015). Malware detection in the cloud under Ensemble Empirical Mode Decomposition. In *2015 Int. Conf. Comput. Netw. Commun.*, pages 82–88. IEEE.

[MAWI] MAWI. Measurement and Analysis on the WIDE Internet. http://www.wide.ad.jp/project/wg/mawi.html. Last Accessed: 2017-02-17.

[57] Mazel, J., Fontugne, R., and Fukuda, K. (2014). A taxonomy of anomalies in backbone network traffic. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International*, pages 30–36. IEEE.

[58] Morton, A. (2016). Active and Passive Metrics and Methods (with Hybrid Types In-Between). RFC 7799, RFC Editor.

[59] Münz, G., Li, S., and Carle, G. (2007). Traffic Anomaly Detection Using K-Means Clustering.

[60] Muraleedharan, N. (2008). Analysis of TCP flow data for traffic anomaly and scan detection. In *2008 16th IEEE Int. Conf. Networks*, pages 1–4, New Delhi. IEEE.

[61] Muraleedharan, N., Parmar, A., and Kumar, M. (2010). A flow based anomaly detection system using chi-square technique. In *Advance Computing Conference (IACC), 2010 IEEE 2nd International*, pages 285–289. IEEE.

[Muuss] Muuss, M. PING. http://ftp.arl.mil/~mike/ping.html. Last Accessed: 2017-02-17.

[63] Namayanja, J. M. and Janeja, V. P. (2014). Change detection in temporally evolving computer networks: A big data framework. In *2014 IEEE Int. Conf. Big Data (Big Data)*, pages 54–61. IEEE.

[64] Ndong, J. and Salamatian, K. (2011). A Robust Anomaly Detection Technique Using Combined Statistical Methods. *2011 Ninth Annu. Commun. Networks Serv. Res. Conf.*, pages 101–108.

[NFCAPD] NFCAPD. Netflow capture daemon. http://nfdump.sourceforge.net. Last Accessed: 2017-02-17.

[NFDUMP] NFDUMP. Netflow collecting and processing tools. http://nfdump.sourceforge.net. Last Accessed: 2017-02-17.

[67] OpenSignal (2016). The State of LTE, 2015 Q4. https://opensignal.com/reports/2016/02/state-of-lte-q4-2015. Last Accessed: 2017-02-17.

[68] Portnoy, L., Eskin, E., and Stolfo, S. (2001). Intrusion Detection with Unlabeled Data Using Clustering. In *In Proceedings of ACM CSS*

Workshop on Data Mining Applied to Security (DMSA-2001), pages 5–8. Citeseer.

[69] Postel, J. (1981). Internet Control Message Protocol. RFC 792, RFC Editor.

[70] Quittek, J. and White, K. (2006). Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations. Technical Report 4560, RFC Editor.

[71] Reynolds, J. (1989). Helminthiasis of the Internet. RFC 1135, RFC Editor.

[72] Ringberg, H., Soule, A., Rexford, J., and Diot, C. (2007). Sensitivity of PCA for traffic anomaly detection. *ACM SIGMETRICS Perform. Eval. Rev.*, 35(1):109.

[73] Samuel, A. L. (1959). Some studies in Machine Learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229.

[74] Scarfone, K. A. and Mell, P. M. (2007). Guide to Intrusion Detection and Prevention Systems (IDPS). Technical Report February, National Institute of Standards and Technology, Gaithersburg, MD.

[75] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V. (2003). RTP: A Transport Protocol for Real-Time Applications. RFC 3550, RFC Editor.

[76] Shirey, R. (2007). Internet Security Glossary, Version 2. RFC 4949, RFC Editor.

[Softflowd] Softflowd. Flow-based Network Traffic Analyser. https://code.google.com/archive/p/softflowd/. Last Accessed: 2017-02-17.

[78] Solaimani, M., Iftekhar, M., Khan, L., and Thuraisingham, B. (2014). Statistical technique for online anomaly detection using Spark over heterogeneous data from multi-source VMware performance data. In *2014 IEEE International Conference on Big Data (Big Data)*, pages 1086–1094, Washington, DC. IEEE.

[79] Soule, A., Salamatian, K., and Taft, N. (2005). Combining filtering and statistical methods for anomaly detection. *Proc. 5th ACM SIGCOMM Conf. Internet Meas. IMC 05*, 4(November):1.

[80] Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A., and Stiller, B. (2010). An Overview of IP Flow-Based Intrusion Detection. *IEEE Communications Surveys & Tutorials*, 12(3):343–356.

[81] Sui Song, Li Ling, and Manikopoulo, C. (2006). Flow-based Statistical Aggregation Schemes for Network Anomaly Detection. In *2006 IEEE Int. Conf. Networking, Sens. Control*, pages 786–791, Ft. Lauderdale, FL. IEEE.

[82] Tierney, B., Kissel, E., Swany, M., and Pouyoul, E. (2012). Efficient data transfer protocols for big data. In *2012 IEEE 8th Int. Conf. E-Science*, pages 1–9. IEEE.

[83] Versi, E. (1992). "Gold standard" is an appropriate term. *BMJ*, 305(6846):187–187.

[84] Wang, G., Ng, T. E., and Shaikh, A. (2012). Programming your network at run-time for big data applications. In *Proc. first Work. Hot*

Top. Softw. Defin. networks - HotSDN '12, page 103, New York, New York, USA. ACM Press.

[85] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: Cluster Computing with Working Sets. pages 10–10.

[86] Zahavi, E., Keslassy, I., and Kolodny, A. (2012). Distributed adaptive routing for big-data applications running on data center networks. In *Proc. eighth ACM/IEEE Symp. Archit. Netw. Commun. Syst. - ANCS '12*, page 99. ACM Press.

[87] Zheng, K., Yang, Z., Zhang, K., Chatzimisios, P., Yang, K., and Xiang, W. (2016). Big data-driven optimization for mobile networks toward 5G. *IEEE Netw.*, 30(1):44–51.

[88] Zseby, T., Molina, M., Duffield, N., Niccolini, S., and F. Raspall, F. (2009). Sampling and Filtering Techniques for IP Packet Selection. RFC 5475, RFC Editor.