Université de Toulouse

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Cotutelle internationale avec :

Présentée et soutenue par :
Mazel Johan
Le lundi 19 décembre 2011

Titre :

Unsupervised network anomaly detection

ED MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture

Unité de recherche :
LAAS-CNRS

Directeur(s) de Thèse :

Owezarski Philippe
Labit Yann

Rapporteurs :
Festor Olivier
Leduc Guy
Fukuda Kensuke

Autre(s) membre(s) du jury :

Chassot Christophe
Vaton Sandrine

Acknowledgements

I would like to first thank my PhD advisors for their help, support and for letting me lead my research toward my own direction. Their inputs and comments along the stages of this thesis have been highly valuable. I want to especially thank them for having been able to dive into my work without drowning, and then, provide me useful remarks.

I want to make a special thank to Pedro Casas-Hernandez who worked with me inside the ECODE project at LAAS-CNRS. Working with him during one year and a half has been a real pleasure. I think that the extent of this thesis contributions would not have been the same without Pedro's work.

I extend a special thank to Ion Alberdi. He introduced me to functionnal programming through the Ocaml language and pointed me toward new and exciting territories in terms of programmation and computer science in general. Our discussions have been on of the most valuable discussions I ever had.

I want to also thank all the people working at LAAS-CNRS for the help they provided me. I here want to especially thank David Gauchard and Christian Berty for their exceptionnal availability and reactivity.

I finally want to thank my family, friends and colleagues at LAAS-CNRS. I won't quote any names but be assured that I am truly grateful for your support during these three and half years.

Abstract

Anomaly detection has become a vital component of any network in today's Internet. Ranging from non-malicious unexpected events such as flash-crowds and failures, to network attacks such as denials-of-service and network scans, network traffic anomalies can have serious detrimental effects on the performance and integrity of the network. The continuous arising of new anomalies and attacks create a continuous challenge to cope with events that put the network integrity at risk. Moreover, the inner polymorphic nature of traffic caused, among other things, by a highly changing protocol landscape, complicates anomaly detection system's task. In fact, most network anomaly detection systems proposed so far employ knowledge-dependent techniques, using either misuse detection signature-based detection methods or anomaly detection relying on supervised-learning techniques. However, both approaches present major limitations: the former fails to detect and characterize unknown anomalies (letting the network unprotected for long periods) and the latter requires training over labeled normal traffic, which is a difficult and expensive stage that need to be updated on a regular basis to follow network traffic evolution. Such limitations impose a serious bottleneck to the previously presented problem.

We introduce an unsupervised approach to detect and characterize network anomalies, without relying on signatures, statistical training, or labeled traffic, which represents a significant step towards the autonomy of networks. Unsupervised detection is accomplished by means of robust data-clustering techniques, combining Sub-Space clustering with Evidence Accumulation or Inter-Clustering Results Association, to blindly identify anomalies in traffic flows. Correlating the results of several unsupervised detections is also performed to improve detection robustness. The correlation results are further used along other anomaly characteristics to build an anomaly hierarchy in terms of dangerousness. Characterization is then achieved by building efficient filtering rules to describe a detected anomaly. The detection and characterization performances and sensitivities to parameters are evaluated over a substantial subset of the MAWI repository which contains real network traffic traces.

Our work shows that unsupervised learning techniques allow anomaly detection systems to isolate anomalous traffic without any previous knowledge. We think that this contribution constitutes a great step towards autonomous network anomaly detection.

This PhD thesis has been funded through the ECODE project by the European Commission under the Framework Programme 7. The goal of this project is to develop, implement, and validate experimentally a cognitive routing system that meet the challenges experienced by the Internet in terms of manageability and security, availability and accountability, as well as routing system scalability and quality. The concerned use case inside the ECODE project is network anomaly detection.

Keywords: anomaly detection, network traffic, unsupervised learning

Résumé

La détection d'anomalies est une tâche critique de l'administration des réseaux. L'apparition continue de nouvelles anomalies et la nature changeante du trafic réseau compliquent de fait la détection d'anomalies. Les méthodes existantes de détection d'anomalies s'appuient sur une connaissance préalable du trafic : soit via des signatures créées à partir d'anomalies connues, soit via un profil de normalité. Ces deux approches sont limitées : la première ne peut détecter les nouvelles anomalies et la seconde requiert une constante mise à jour de son profil de normalité. Ces deux aspects limitent de façon importante l'efficacité des méthodes de détection existantes.

Nous présentons une approche non-supervisée qui permet de détecter et caractériser les anomalies réseaux de façon autonome. Notre approche utilise des techniques de partitionnement afin d'identifier les flux anormaux. Nous proposons également plusieurs techniques qui permettent de traiter les anomalies extraites pour faciliter la tâche des opérateurs. Nous évaluons les performances de notre système sur des traces de trafic réel issues de la base de trace MAWI. Les résultats obtenus mettent en évidence la possibilité de mettre en place des systèmes de détection d'anomalies autonomes et fonctionnant sans connaissance préalable.

Mots-clés : détection d'anomalies, trafic réseau, apprentissage non-supervisé

Contents

Chapter 1

Introduction

Anomaly detection in networks has constituted a constant interest for every player of the networking game. From Internet Service Provider (ISP) to security researchers, many players have been trying to find efficient techniques to detect network anomalies. Network anomalies can be defined as events that represent a deviation from the normal state of the network. On one hand, security researchers find interests in anomalies, such as scans that can sometimes be viewed as signs of incoming attacks such as Denial of Service (DoS) or Distributed Denial of Service (DDoS). On the other hand, ISPs see anomalies, such as DoS, as threats to their day-to-day operations since they can impact customers experience or even worse, disrupt the offered service. The attack targeting Georgia's IT systems during the summer 2008 proved how vulnerable such Information technology (IT) systems are to DoS attacks [1]. A more recent example of such events is the "Avenge Assange" campaign that took place in December 2010. This campaign was organized by a group of online activists calling themselves Anonymous. It targeted several corporations that refused to support donations for the Wikileaks website.[2] , [3] Anomalies can thus be roughly defined as defined as events that leads a part of network trafic to deviate from its normal or usual state. These examples clearly motivate the need to detect such events.

Several techniques have been used to detect such events. The first one is host anomaly detection. In this case, a detection system analyzes the behavior of an host and tries to assess the abnormality of its actions. The second type of detection systems is network anomaly detection systems. These systems instead rely on network traffic analysis in order to decide if a specific part of network traffic is anomalous or not. We focus our work on this particular type of detection systems.

Network anomaly detection systems however face many problems. The main challenge in automatically detecting traffic anomalies is that these are moving targets. In fact, it is extremely difficult to precisely and permanently define the set of possible anomalies that may arise, especially in the case of network attacks, because new attacks as well as new variants to already known attacks are continuously emerging. This problem is even worsened by the fact that anomalies appear inside a traffic landscape that is itself changing. Network traffic is in fact continuously evolving under many influences: increase in volume, topology changes, changes in application uses, etc.

[1]http://www.nytimes.com/2008/08/13/technology/13cyber.html
[2]http://www.guardian.co.uk/media/2010/dec/08/operation-payback-mastercard-website-wikileaks
[3]http://pandalabs.pandasecurity.com/operationpayback-Broadens-to-operation-avenge-assange/

This dissertation aims at designing a general anomaly detection system that would be able to detect a wide range of anomalies with diverse structures, using the least amount of previous knowledge and information, ideally none. The remaining of this chapter is composed of two sections. The first one introduces the different reasons that lead us to think that network anomaly detection systems needs to be more autonomous and how we plan to achieve this goal. The second section of this chapter present our contributions towards the design of such systems.

1.1 Motivating problems

There are several motivations for the introduction of knowledge-independent network anomaly detection systems. These reasons are presented in the next two sections. First, legitimate and illegitimate network traffics display several types of changes along time and space. Second, current anomaly detection systems experience difficulties to cope with these behaviors. The next two sections address these topics.

1.1.1 Changing traffic in an evolving Internet

We address the mutating nature of traffic from two points of view: the changes of network traffic in general and the changes in terms of network anomalies.

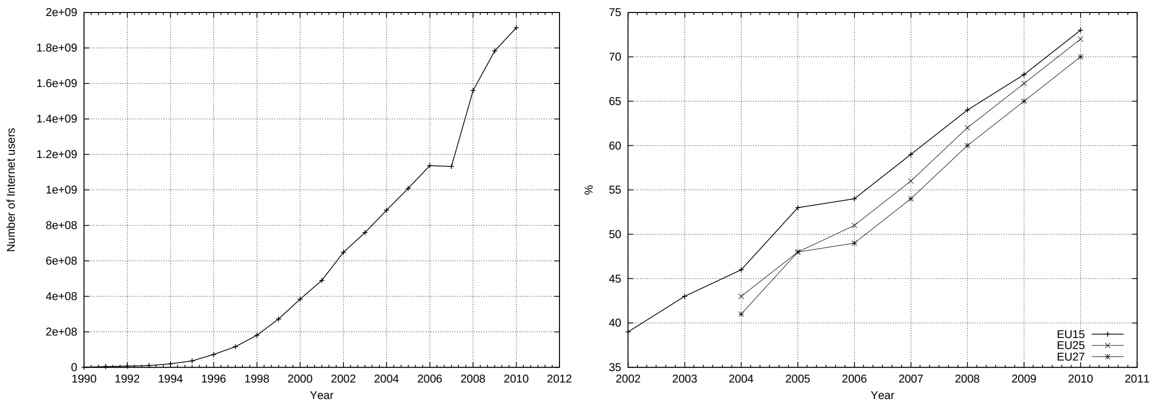
1.1.1.1 Changing traffic

Network traffic is changing. These changes can be classified into three distinct phenomenons. We here provide a short taxonomy of these phenomenons. First, network traffic evolves along time, and according to several time-scales, concerning its volume in terms of number of packet and number of bytes. We name this characteristic the traffic evolution. The second phenomenon is traffic changes due to “spatial” variation. Traffic at the ingress router of an Autonomous System (AS) is different than traffic in a backbone link. Traffic also varies between two physical links or regions (country, continent, etc.). We call this phenomenon the traffic disparity. Third, traffic inner nature changes over time through the creation or mutation of applications, and the emergence of new data exchange paradigm. We call this phenomenon traffic mutability. The next paragraphs detail these three phenomenons.

- The first evolving property of Internet network traffic is its variation in terms of volume along time. This variation can be divided into two distinct sub-phenomena: first, the cyclic oscillations, and second, the ever-increasing volume. The first of these sub-phenomenons is the daily and weekly variation of traffic. In [1], Roughan et al. study Simple Network Management Protocol (SNMP) data from one of the largest operational Internet backbones in North America (AT&T). In [2], Papagiannaki et al. process SNMP measurements from an IP backbone network. In [3], Cho et al. analyze both aggregated traffic from several Japanese Internet Service Providers (ISPs) and detailed records on a single ISP. All works quoted above observe the previously presented daily and weekly variations.

The second of these sub-phenomenons is the constant increase of traffic in terms of volume. We investigate this by looking at network traffic of the Wide repository [4]. We consider traffic captured at samplepoints B and F that are located on a trans-pacific transit link between Japan and the United States of America. We are thus able to assess the bandwidth increase on this particular link. A simple analysis reveals that, between 2001 and 2008, the average throughput evolved from approximately 10Mbps in January 2001 to more than 250Mbps in September 2011. This demonstrates the increases in terms of traffic increase along time. In [2], Papagiannaki et al. observe that, for 2 out of 3 Points of Presence (PoPs), traffic volume increase from October 2000 to July 2002. Traffic volume in the third Point of Presence (PoP) remains constant over the years. In [5], Cho et al. shows that traffic at Japanese major Internet exchange points has constantly increased over the last ten years. From 2005 to the start of 2008, network traffic has roughly doubled every two years. In [6], the Cisco company declares that the traffic increased eightfold over the last five years.

Two main reasons of this constant growth can be identified. On one hand, the number of Internet users constantly increase. Figure 1.1 [4] , [5] clearly exposes this trend over the years.



(a) (b)

Figure 1.1: Number of internet users in the world (a) and proportion of households connected to Internet in Europe (b) (UE15 stands the 15 European Union members between 1995 and 2004, UE25 for the 25 members between 2004 and 2007 and UE27 for the 27 members between 2007 and up to now).

On the other hand, a race towards a greater bandwidth available for the end-user is happening as we speak. In [5], Cho et al. show that the number of Fiber-To-The-Home (FTTH) subscribers and cable TV subscribers in Japan have constantly increased from 2000 to 2008. The number of Asymmetric Digital Subscriber Line (ADSL) subscribers displays a steeper increase but reach a peak approximately located at the start of 2006 and shows a slow decay afterwards. Figure 1.2 [6] displays

[4]http://data.un.org/Data.aspx?q=internet&d=MDG&f=seriesRowID%3a608
[5]http://epp.eurostat.ec.europa.eu/portal/page/portal/information_society/data/
[6]http://epp.eurostat.ec.europa.eu/portal/page/portal/information_society/introduction

the broadband penetration rate per inhabitant and household for three sets of European Union member (cf. caption). Figure 1.3 [7] displays the increase in terms of number of broadband users worldwide. Both these figures show that the number of broadband connections increases along time in Europe and thus causes an increase in terms of traffic.

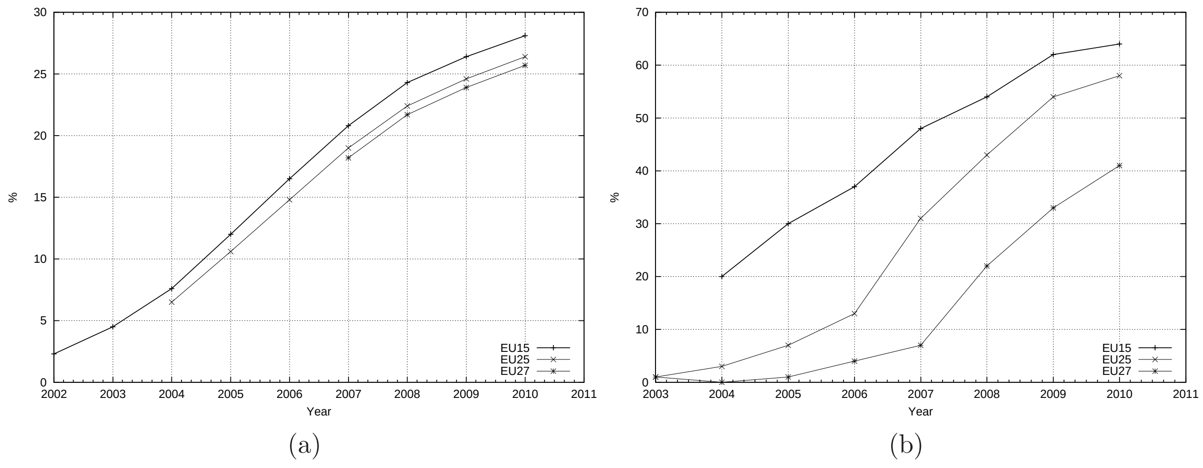


Figure 1.2: Broadband penetration rate in Europe for inhabitant (a) and households (b) (UE15 stands the 15 European Union members between 1995 and 2004, UE25 for the 25 members between 2004 and 2007 and UE27 for the 27 members between 2007 and up to now).

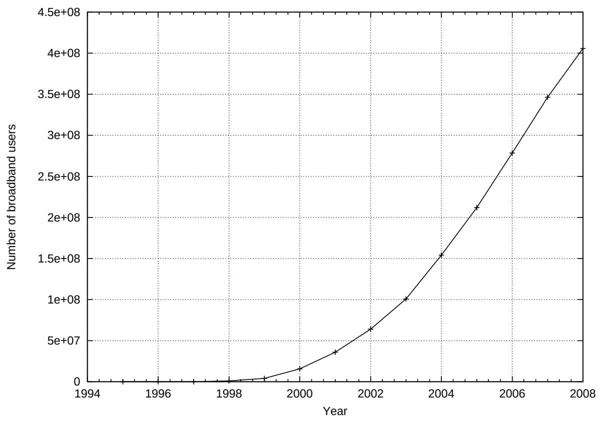


Figure 1.3: Number of broadband users in the world.

This demonstrates that the traffic volume has constantly increased over the last ten years.

- The second phenomenon exposed in our taxonomy is disparity. It names the difference of traffic when it is observed from different physical points of view. In [7], Ringberg et al. expose the difference in terms of average number of packets by flows between traffic aggregated by input links, ingress routers or Origin-Destination

[7]http://data.worldbank.org/indicator

flows. They also advances that traffic variability is highly changing. In [2], Papagiannaki et al. find that the traffic displays different evolutions between three PoPs: traffic increases in two PoPs while it remains constant for another one. Such difference in terms of temporal evolution implicitly confirms that there are difference in of traffic nature between these PoPs.

- Network traffic, in its inner nature, has been highly changing along Internet evolution. As exposed before, we call this phenomenon traffic mutation. We illustrate this through two examples. First, the emergence of new protocols or applications that modify the shape of traffic. And second, the impact of civil society over networks through legislation.

  The first factor that causes change in Internet traffic is the emergence of new protocols or services. Their progression can be extremely fast, unpredictable and impacting on traffic structure. For example, the Netflix Video on Demand (VoD) service in Canada is a good illustration of this fact. It has been launched on September 22nd 2010 and has experienced a constant growth of its number of users [8]. By the end of march 2011, Netflix traffic consumes 13.5 percents of downstream traffic during evening peak hours in Canada [9]. This percentage of traffic has appeared in only 6 months. This illustrates the highly changing nature of network traffic composition on a relatively short time scale. On a bigger scale and for information purpose, Netflix accounts for almost a quarter of total bytes and nearly 30% of the downward traffic of peak period in North America traffic [9]. Another example of the changing protocol landscape is the constant increase in use of web radios and web televisions. Figure 1.4 [8] shows the increase of the number of individuals using Internet to listen web radios or watch web televisions. This proves that the protocols associated these (relatively new) uses have been and continue to grow thus changing the protocol landscape.

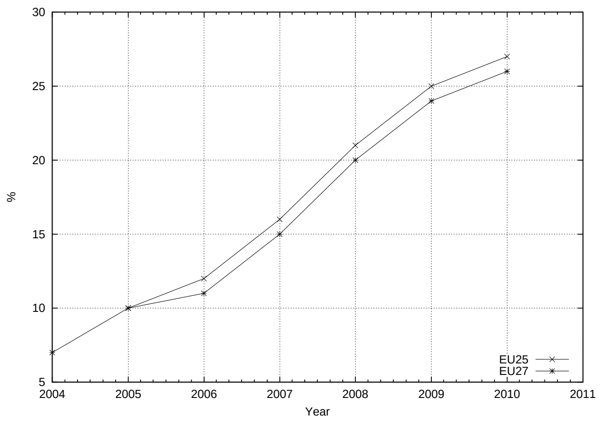


Figure 1.4: Percentage of individuals using the Internet for listening to web radio or watching web television (UE25 for the 25 members between 2004 and 2007 and UE27 for the 27 members between 2007 and up to now).

The exchange of copyrighted data and the rise of the associated techniques is another example of the fast mutation of network traffic. Up to now, these exchanges have

[8]http://epp.eurostat.ec.europa.eu/portal/page/portal/information_society/data/

been using peer-to-peer protocols. The first episode of this phenomenon is the rise and fall of Napster between June 1999 and July 2001. [9] Its lifetime was short. But it quickly represented a significant share of traffic since its traffic was close to Hypertext Transfer Protocol (HTTP) traffic in terms of volume in certain conditions [10].

Peer-to-peer protocols can also be used as example for the second cause for the changing nature of traffic: the direct impact of the civil society over networks through legislation. In fact, in the Napster case, illegal file exchange of copyrighted material through its infrastructure initiated a series of lawsuits that directly caused the end of Napster. It is the very centralized nature of Napster that caused its death by offering a single easy target. The pressure over peer-to-peer traffic and protocols then directly induced the mutation of peer-to-peer protocols toward a new decentralized paradigm illustrated by Gnutella-like protocols [11]. As a consequence, port-based traffic classification tools were introduced to identify this second generation of peer-to-peer protocols through port number profiles. A third generation of peer-to-peer protocols then emerged to cope with port-based traffic classification. These protocols use random ports in order to avoid detection [11, 12]. During the past years, several countries have introduced new laws (Loi Création et Internet in France [10] and amended Copyright Act in Japan [11]) in order to drastically reduce the exchange of copyrighted data through peer-to-peer protocols. The probable consequence of these events has been a reduction of peer-to-peer traffic exchange and an increase of HTTP traffic from one-click file hosting websites to end-users [13]. This evolution is corroborated by the fact, in [14], Maier et al. analyze ADSL traffic and find that 15% of HTTP traffic is composed of RAR archive files that can be suspected of containing copyrighted data. During the night, this proportion increases as every other HTTP traffic components' proportions decrease [14]. Furthermore, in [15], the Cisco company says that P2P file sharing decreases its worldwide share of overall traffic volume from 38% to 25% which represents a decrease of 34% percents. Peer-to-peer protocols and traffic followed a complicated path with multiple deep changes in their structure. Their use seems even to decrease under pressure from legislations.

1.1.1.2 Mutating anomalies

Network anomalies can be of very different types. Some anomalies such as scans are mostly harmless by themselves. They however are generally signs of a possible incoming attack. On one hand, host scans target a single machine. On the other hand, network scans aim at either mapping a particular network and acquire knowledge about its structure or finding machines sharing a particular vulnerability. Other anomalies such as floods or DoS/DDoS aim at degrading a particular service on a restrained set of targets. They saturate their target(s) by sending a huge number of packets that effectively forbids legitimate users to access the considered service. These anomalies are sometimes called attacks because of their deliberate intent to harm their target(s).

[9]http://www.businessweek.com/2000/00_33/b3694003.htm
[10]http://www.legifrance.gouv.fr/
[11]http://www.mext.go.jp/b_menu/houan/an/171/1251917.htm

Similarly to traffic itself, these anomalies also exhibit a mutating behavior. In their longitudinal analysis of the MAWI [4] traffic repository [12], Borgnat et al. state that some anomalies, like SYN scans and floods are recurrent over the years. However other anomalies are very localized in the time line of the repository: Network News Transfer Protocol (NNTP) anomalies during the first years, Secure Shell (SSH) since 2004 and Microsoft security holes from August 2003. Furthermore, some anomalies can be accurately located in time and sometimes represent the majority of network traffic. One can here quote two events. First, a continuous Ping flood from August 2003 to December 2003 represented more than 50% of traffic volume from Japan to the USA in terms of number of packets. Second, several outbursts of traffic due to the Sasser worm in August 2004, December 2004 and march 2005 accounted for more than 50% of traffic volume from the USA to Japan regarding the number of packets. The constant worm emergence, from the Sasser worm, and many before, to the latest example of spreading worm, the Morto Worm [12] constitutes another example of anomaly mutations. All these examples demonstrate that anomalies are always present in traffic and mutate along time. In [16], Allman et al. study scans targeting the Lawrence Berkeley National Laboratory (LBNL) network from June 1 1994 to December 23 2006. The authors show that scanning greatly evolved along the years. All great increases in terms of scanning connections coincide with the emergence of new worms. The increase in 2001 reflects the appearance of CoreRed and Nimda worms. The scan surge of 2004 seems to be linked with the emergence of MyDoom, Sasser, Welchia, Bobax and Gaobot worms. The service or port number also evolves along time. For example, the 9898 port starts to be targeted when the Sasser worm appears. This is coherent since port 9898 was a port number used by the Sasser worm as backdoor.

Congestions and anomalies also have an impact on traffic characteristics. For example, they modify the protocol mix which influence marginal distributions of aggregated packet and byte count and long-range dependence possibly estimated through Hurst parameter [12]. Since the appearance and nature of such phenomenons are not predictable, traffic characteristics for the considered statistical indexes are corollary unpredictable. This impact over traffic further increases the traffic mutation phenomenon explained above.

1.1.2 Weaknesses of current network anomaly detection approaches

The problem of network anomaly detection has been extensively studied during the last decade. Two different approaches are by far dominant in current research literature and commercial detection systems: signature-based and anomaly detection. Both approaches require some kind of guidance to work, hence they are generally referred to as knowledge-based approaches. Signature-based detection systems, such as Bro and Snort, are highly effective to detect those anomalies which are programmed to alert on. When a new anomaly is discovered, generally after its occurrence, the associated signature is coded by human experts, which is then used to detect a new occurrence of the same anomaly. Such a detection approach is powerful and very easy to understand, because the operator can directly relate the detected anomaly to its specific signature. However, these systems

[12]http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=Worm%3AWin32%2FMorto.A

cannot defend the network against new anomalies, simply because they cannot recognize what they do not know. Furthermore, building new signatures is expensive, as it involves manual inspection by human experts.

On the other hand, anomaly detection detects anomalies as patterns that deviate from a previously built model [17, 18]. Such methods can detect new kinds of anomalies not seen before, because they will naturally deviate from the baseline. In order to build their model, anomaly detection either requires fine tuning in order to only captures anomalous events, or supervised learning to automatize model building step as in [19]. This training however depends on the availability of purely anomaly-free traffic data-sets. Labeling traffic as anomaly-free is expensive and hard to achieve in practice, since it is difficult to guarantee that no anomalies are hidden inside the traffic. Additionally, it is not easy to maintain an accurate and up-to-date model for anomaly-free traffic, particularly when new services and applications are constantly emerging (cf. section 1.1.1). However, provided that anomaly free traffic is available, supervised-learning and anomaly detection-based system provides a first solution to cope with the changing nature of traffic.

Supervised-learning has also been applied to signature-based detection systems. It uses labeled anomalies to train a model for anomalous traffic as in [20]. Such systems however suffer from the same weaknesses of signature-based models. They cannot recognize unknown events. They however provide an automated technique to produce anomaly signatures.

Apart from detection, and in order to take accurate countermeasures, operators need to analyze and characterize network anomalies. However, network anomaly detection systems often only give an alarm of an occurring anomaly without any further description of the occurring anomaly. These systems force operators to proceed to an hard and time-consuming task of anomaly characterization. The analysis may become a particular bottleneck when new anomalies are detected, because the network operator has to manually dig into many traffic descriptors to understand its nature. The high number of alarms and hence possible anomalies can also increase the amount of work. In current traffic scenario, even expert operators can be quickly overwhelmed if further information is not provided to prioritize the time spent in the analysis.

1.1.3 The need for an autonomous anomaly detection

The original hypothesis used in anomaly detection is:

Hypothesis 1.1 *The anomalous traffic is statistically different from normal traffic. [21, 22]*

As we stated in section 1.1.2, current anomaly detection systems use two main detection approaches: signature-based or misuse detection and anomaly detection. Both of these approaches use hypothesis 1.1 and build either signatures on anomalous traffic (signature-based detection) or normal traffic (anomaly-based detection). However as we showed in section 1.1.1, network traffic exhibits several type of changing behaviors, evolution, disparity and mutability. Current anomaly detection systems have troubles dealing with these phenomenons.

We therefore think that anomaly detection needs to cope with the changes in both anomalous and normal network traffic. Anomaly detection system shall not use detailed knowledge about traffic or anomaly structure. They shall instead only use a single hypothesis to separate anomalous traffic from normal one:

Hypothesis 1.2 *The majority of traffic representation instances is normal. Which is equivalent to: only X% of traffic representation instances are malicious with $X < 50\%$. [23]*

Efficient anomaly detection systems should therefore use hypothesis 1.1 and 1.2 in order to autonomously find anomalies. Unsupervised learning techniques offers us tools to create classes of similar patterns and isolate classes of patterns different from each other without previous knowledege on patterns. These techniques therefore allow anomaly detection systems to separate anomalous traffic from normal one provided that hypothesis 1.1 is verified. Once anomalous and normal traffic are separated, hypothesis 1.2 provides us with a criterion to identify normal traffic and by extension anomalous traffic.

This unsupervised approach allows us to autonomously extract abnormal traffic without any predefined models of either normal or abnormal traffic. However, this approach only work provided that both hypothesis are verified. Hypothesis 1.1 is a classic hypothesis of the network anomaly detection field. While 1.2 has never been used in this context. This approach also a need an unsupervised technique that will be able to efficiently performs the extraction. It is only the final evaluation of our tool that will enable us to assess the pertinence of the hypothesis and the efficiency of the chosen unsupervised method.

We think that unsupervised anomaly detection is the scheme to follow in order to build knowledge-independent and robust anomaly detection systems regarding network traffic evolution, disparity and mutability.

1.1.4 Summary

Normal and abnormal network traffics change over time and across measuring point. We state that current anomaly detection approaches relying on a one-time-built model for legitimate or illegitimate traffics is inefficient regarding these phenomena. The building and maintenance costs of these models further confirm that these techniques should be avoided. We therefore think that modern anomaly detection systems should be able to autonomously cope with traffic evolution, disparity and mutability through unsupervised learning. Such use of unsupervised learning allows network anomaly detection systems to be immune to these phenomenoms at the cost of using hypotheses on network traffic.

1.2 Contributions

Our contribution is twofold. The first part is an anomaly detection method relying on unsupervised learning. The second part is a post-processing step that correlates, ranks and characterizes found anomalies.

1.2.1 Unsupervised anomaly detection

In the first part of the thesis, we first propose several techniques to process traffic and generate meaningful attributes. We then describe how anomalies behave for the built attributes provided that anomalies are statistically different from normal traffic (cf. hypothesis 1.1). We finally propose a robust method that relies on unsupervised learning to discover traffic flows containing network traffic of similar nature. By using hypothesis 1.2, we are able to determine the normal traffic and corollary the potentially anomalous one. Our method is based on clustering techniques and especially subspace-clustering. This method allows us to blindly find anomalous traffic while using the two hypothesis quoted above.

1.2.2 Anomaly post-processing

We present several methods to process results of anomaly mining.

We first propose a correlation technique in order to reduce the number of alarms. In order to do so, we use the sets of source and destination of an anomaly as its unique identifiers. We then use the similarity between source and destination of anomalies in order to assess anomaly similarity.

After the correlation, we introduce a dangerousness assessment metric. This metric allows us to rank anomalies, easily discriminating harmless anomalies from dangerous one.

We also propose a method to characterize previously mined anomalies. This method is able to build signatures that isolate anomalous traffic from normal one. This step is critical in the sense that it allows operators to easily understand the considered anomalous traffic nature. The built signature also allows operators to quickly cope with the anomaly through deployment of the considered signature on any network security device (Intrusion Detection Systems (IDS), Intrusion Protection Systems (IPS), firewall, etc.) and hence greatly improves the global autonomy of our system.

1.2.3 Summary

Our contribution is composed of two main components. The first part is an unsupervised technique to detect anomaly relying on subspace clustering, anomaly correlation and anomaly ranking. The second part is a post-processing step technique which allows us to correlate, rank and characterize anomalies.

1.3 Summary

Network anomaly detection is a critical component of modern network management systems used by ISPs. Anomaly detection systems use one-time-built and expensive-to-build models to assess the presence of an anomalous event. Current anomaly detection systems are therefore extremely vulnerable to novelty which is an inner characteristics of network traffic as we show in section 1.1.1. We therefore think that anomaly detection should use the hypothesis presented in the section 1.1.3 and unsupervised learning tools in order to isolate anomalies. We think that our proposition constitutes a major first step toward autonomous network anomaly detection.

1.4 Context inside the ECODE project

This thesis has been realized in the framework of the ECODE project. This project is funded by the European Commission under grant FP7-ICT-2007-2/223936. It aims at introducing cognitive capacities in routing systems. The partners are: Alcatel-Lucent BELL, Université Catholique de Louvain, Université de Liège, the Interdisciplinair instituut voor BreedBand Technologie (IBBT), the Institut National de Recherche en Informatique et en Automatique (INRIA), Lancaster University (ULANC) and the Centre National de la Recherche Scientifique (CNRS).

First, we present the goals of the ECODE project. Second, we address use cases inside the project. Finally, we present the proposed architecture for future routing systems.

1.4.1 Overview

The goal of the ECODE project is to develop, implement, and validate experimentally a cognitive routing system that can meet the challenges experienced by the Internet in terms of manageability and security, availability and accountability, as well as routing system scalability and quality. By combining both networking and machine learning research fields, the resulting cognitive routing system fundamentally revisits the capabilities of the Internet networking layer so as to address these challenges altogether.

For this purpose, the project investigates and elaborates novel, on line, and distributed machine learning techniques kernel of the cognitive routing system. During the building phase, the cognitive routing system is both designed and prototyped. In the second phase, three sets of use cases are experimented to evaluate the benefits of the developed machine learning techniques. The experimentation and the validation of these techniques are carried out on physical (iLAB) and virtual (e.g. OneLab) experimental facilities.

1.4.2 Use cases

The ECODE project targets all the routing problematics. Those are: adaptive traffic sampling and management, path performance monitoring, intrusion and attack/anomaly detection, path availability, network recovery and resiliency, profile-based accountability, routing system scalability and quality. Our work is part of the intrusion and attack/anomaly detection use case. Our role as an ECODE partner was to devise new and innovative ways to use cognitive techniques such as supervised, semi-supervised or unsupervised machine learning in our field, i.e. network anomaly detection.

1.4.3 Proposed architecture

Figure 1.5 exposes the paradigm change proposed by the ECODE project concerning routing systems. The rationale is to improve existing routing and control lower-level data collection and decision making with a cognitive engine. This cognitive engine would be able to enable systems and network to learn about its own behavior and environment over time and analyze problems, tune its operation and increase its functionality and performances.

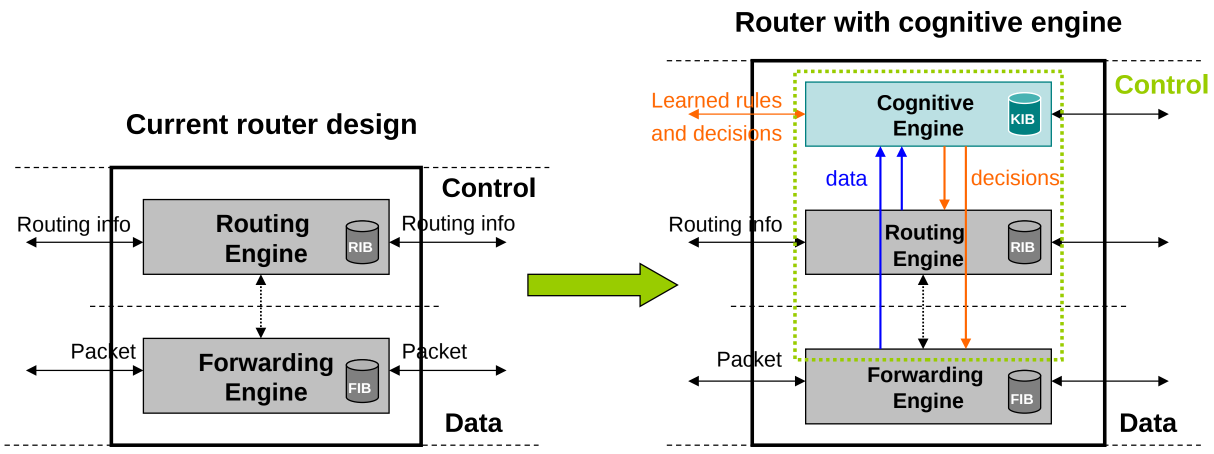


Figure 1.5: High-level architecture of proposed ECODE routing system.

1.4.4 Summary

This thesis has been lead inside the ECODE project. This project is financed by the European Commission. Its goal is to introduce machine learning capabilities inside routing systems in order to improve their autonomy. It targets the whole functionality sets of routing systems including network anomaly detection.

1.5 Dissertation outline

This dissertation is composed of five chapters:

- Chapter 2 addresses related work in unsupervised learning, network anomaly detection and intrusion detection fields.
- Chapter 3 addresses the design of our unsupervised learning-based anomaly detection method.
- Chapter 4 exposes several techniques that process the results of our detection method in order to ease the work of the network operator.
- Chapter 5 presents an evaluation of our algorithm on real traces.
- Chapter 6 concludes this dissertation by summarizing our contributions and proposing several ideas to improve our proposal in the future.

Chapter 2

Related work

This chapter introduces some related work previously done in the fields of unsupervised machine learning and network anomaly and intrusion detection. The first section introduces machine learning methods dedicated to unsupervised learning. The second section then addresses network anomaly detection systems and intrusion detection systems. These systems can be classified in two categories: knowledge based detection systems, either using misuse detection or anomaly detection, and knowledge-independent or unsupervised detection systems. Another classification of network anomaly detection systems and intrusion detection systems is based upon the use, or lack of use, of machine learning technique. We will use this second distinction to present network anomaly detection systems and intrusion detection systems.

Note: references are presented by chronological order inside each section or item.

2.1 Unsupervised machine learning

Unsupervised machine learning aims at exposing structure inside data. Several techniques are commonly used in literature in order to perform unsupervised learning. A non exhaustive list of these techniques is: blind signal separation, neural network models and clustering. We will present blind signal separation, neural network models and clustering in the next two sections. We put a special emphasis the technique that we chose: clustering.

2.1.1 Blind signal separation

Blind Signal Separation (BSS) consists in extracting a set of signals from a set of mixed signals with very few or no information on the nature of the signals. The standard practical use case of BSS is the cocktail party problem where one wants to extract the speech of several persons in the room from several observations. Methods that can be used for this goal include Independent Component Analysis, Principal Components Analysis, etc. Independent Component Analysis (ICA) makes the assumption that signals to isolate are independent. There is two main independence definitions: through minimization of mutual information or through the maximization of non-Gaussianity. Principal Component Analysis (PCA) [24, 25] is a mathematical procedure that maps a set of points to a new sets of axis. The underlying idea is that many of the original axis are correlated.

The method therefore aims at finding a reduced set of uncorrelated axis. The axis that PCA extracts are called principal axes or principal components. These obtained principal components are oriented according to the maximum variance direction in the data. The principal components obtained by PCA are ranked by the amount of variance that they capture. Furthermore, the set of principal components form an orthogonal basis. By using PCA, one is thus able to perform dimensional reduction and project data in a more meaningful space.

2.1.2 Neural networks

Neural networks are also used to perform unsupervised learning. The term neural network actually has two different meanings: biological neural networks and artificial neural networks. Biological neural networks are constituted of real biological neurons in the nervous systems. Artificial neural networks are mathematical or computational models that mimic the structure and/or functional aspects of biological neural networks. An artificial neural network is constituted of interconnected artificial neurons. Artificial neural networks are thus able to model relations between inputs and outputs and to find patterns in data. Among artificial neural network models, one can quote Self-Organizing Maps (SOM) that build a discretized representation of the input space. Adaptive Resonance Theory (ART) has also been used. ART is mainly used in form recognition. Its principle is to compare expected observation to real input. If the input is too far from the expected pattern according to a "vigilance parameter", a new expected observation is added. In our particular use case, a partition could be obtained by using clustering algorithms over SOMs as presented in [26]. This partition would separate abnormal traffic from normal one.

2.1.3 Clustering algorithms

Clustering algorithms aim at grouping instances similar between each other according to a similarity measure between instances. Our presentation of clustering is split in two parts. First, we will present several classic clustering algorithms. Then we will specifically address the challenges of clustering high-dimensional data and the algorithms designed to cope with these problems.

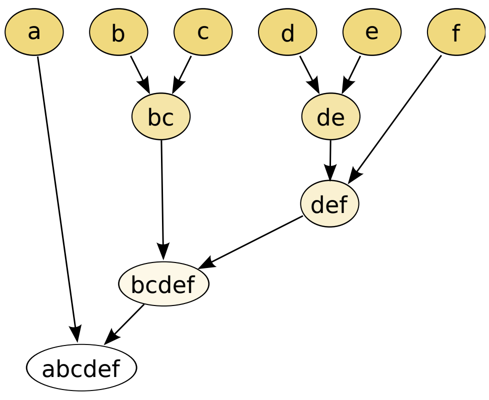
2.1.3.1 General purpose clustering algorithms

Clustering algorithms can be roughly classified into two main categories: hierarchical clustering and partitional clustering.

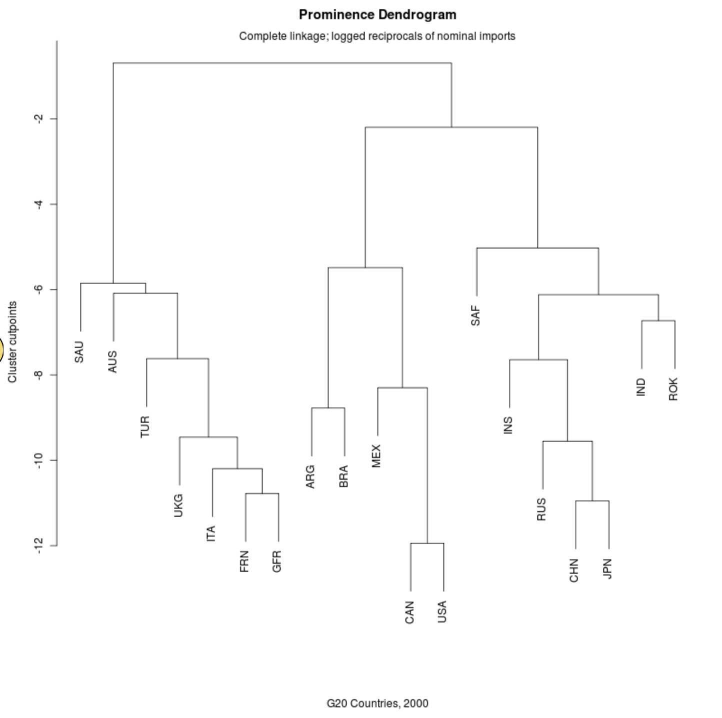
Hierarchical clustering aims at building a hierarchy between clusters. Two strategies are to be used: either agglomerative or divisive. Agglomerative hierarchical clustering uses a bottom-up approach: each instance or group of instances is successively aggregated with its closest neighbor. Divisive hierarchical clustering is a top-down strategy. At the beginning of the procedure, the whole set of instances forms a single cluster [1] that is separated into smaller instances down to each individual instance. The usual representation of the final result of these aggregative or divisive strategy is dendrogram (cf. Figure

[1]A cluster is a group of instances similar according to a similarity measure.

2.1 [2] and [27]). The dendrogram in Figure 2.1b is built by successively grouping similar atomic instances (e.g. the grouping between Canada (CAN) and USA), or grouping atomic instances similar with previously grouped instances (e.g. the grouping between Russia (RUS) on one hand, and China (CHN) and Japan (JPN) on the other hand). The final partition of instances is built over the dendrogram by simply keeping the aggregated instances at a specified level.



(a)



(b)

Figure 2.1: Examples of dendrogram obtained from hierarchical clustering.

Partitional clustering is designed to build partitions on space in a single pass, i.e. separate instances groups according to a similarity measure between instances. Many algorithms have been proposed. Among these algorithms, one can quote k-means clustering algorithm [28] which aims at partitioning instances into a previously fixed number of cluster k. Each instance is associated with a cluster center. The underlying hypothesis of k-means is that every cluster is a circle or an ellipse (depending on the used distance and property of data (normalization, zero-mean, etc.)). DBSCAN [29] is another clustering algorithm that relies on the concept of density. Its parameters are the minimal number of points in a cluster and the distance that defines the density threshold.

2.1.3.2 Clustering data in high-dimensional space

Clustering in high-dimensional data is a special use case of clustering where data to be processed contains many dimension ranging from several dozen to thousands. This amount of dimensions creates a new set of challenges called the "curse of dimensionality".

[2]http://en.wikipedia.org/wiki/File:Hierarchical_clustering_simple_diagram.svg

According to Kriegel et al. [30], this so-called curse is actually constituted of 4 main different problems.

- Multiple dimensions create problems very hard to comprehend. The high number of dimensions makes it impossible to directly tabularize, visualize or enumerate the whole feature space.
- Distance and neighborhood meaning decrease with the increase in dimensionality. The relative distance of the farthest point and the nearest point converges to 0 when the dimensionality d increases. We thus have:
  $$\lim_{d\to\infty} \frac{dist_{max} - dist_{min}}{dist_{min}} \to 0$$
  In other words, when one consider a point, the discrimination between the closest and the farthest neighbor become more and more tenuous as the number of considered dimensions increases.
- A cluster groups objects that are related regarding their attribute's values. However, the relevance of each attribute may vary for each possible cluster. This problem is known as the "local feature relevance": different clusters might be found in different subspaces, so that a global attribute filtering is inefficient.
- High dimensional data means a high number of attributes. Therefore, it is likely that some attributes or some sets of attributes are correlated. Hence, clusters might exist in arbitrarily oriented subspaces.

In order to cope with these challenges, several techniques have been introduced. Kriegel et al. classify techniques for high-dimensional data in three categories: clustering in axis-parallel subspaces, correlation clustering and pattern-based clustering. Clustering in axis-parallel subspaces aims at finding clusters in axis-parallel subspaces. Correlation clustering intends to find clusters in arbitrary oriented subspaces. Pattern-based clustering targets clusters constituted of rows and/or columns. Each of these categories are presented in each of the following parts.

Axis-parallel clustering Clustering in axis-parallel subspaces can itself be separated into four subcategories: projected clustering, soft-projected clustering, subspace clustering and hybrid clustering. Kriegel et al. and Parsons et al., in [31], also separate top-down and bottom-up algorithms. Top-down algorithms try to find clusters from the whole feature space while bottom-up algorithms first search clusters in low-dimensional space and aggregate them to form global clusters. We will not use this distinction to classify algorithms of this category and just use the four subcategories listed above.

The first subcategory is projected clustering. As stated in [30], projected clustering aims at "finding an unique assignment of points to subspace clusters". Noise is also taken into account by some algorithms. They use a dedicated distance function in a clustering procedure applied on the whole space. PROCLUS [32] (improved later by FINDIT [33] and SSPC [34]) and PreDeCon (subspace PREference weighted DEnsity CONnected clustering) [35] are all projected clustering algorithms. PROCLUS is a k-medoid-like [3] clustering algorithm. In the clustering phase, the subspace of each medoid is determined by minimizing the standard deviation of the distance of points in the neighborhood of the medoids to the corresponding medoid for each dimension. The algorithm is able to detect noise as points too far from medoids. FINDIT employs several heuristics that improve efficiency and accuracy. SSPC adds further improvements that enhance accuracy by using domain knowledge in the form of labeled objects and/or labeled attributes. PreDeCon applies DBSCAN to the whole feature space. However, it uses a specialized distance that captures the subspace of each cluster. In order to do so, the algorithm defines a subspace preference for each point that is determined through the variance of the points located in the neighborhood of the considered point.

The second subcategory is soft-projected clustering. It presents a k-means-like approach that takes as parameter the expected number of clusters. It then tries to optimize a function that guarantees a pertinent clustering result regarding the specified number of clusters by adjusting the attribute weights. The Locally Adaptive Clustering (LAC) [36] and Clustering Objects in Subsets of Attributes (COSA) [37] are both soft-projected clustering methods. LAC aims at finding k centroids and k sets of d weights for each cluster (d being the number of dimensions). The algorithm approximates k clusters by adjusting the centroids and weights. COSA uses an approach similar to PreDeCon. It aims at building a matrix that contains, for each point, a subspace preference. A simple clustering can then be used on the matrix to build the global partition.

The third subcategory is subspace clustering. In [30], Kriegel et al. defines subspace clustering as “aiming at finding all clusters in all subspaces of the entire feature space”. The CLIQUE method [38] is the pioneering work in this subcategory. It uses a grid-based algorithm and proceed in a bottom-up fashion to aggregate dense unit. A dense unit is a part of the feature space, for one or several attributes, defined by the grid, that is dense regarding a threshold. Figure 2.2a displays such a grid. One can notice that each cell of this grid has a fixed size. We thus call grids used by CLIQUE, fixed grids. MAFIA [39] is an incremental version of CLIQUE that uses adaptive grids (cf. Figure 2.2b). In this type of grid, each cell dimensions are determined by analyzing density along dimensions. Each cell therefore contains an homogeneous density that is different in its neighbor(s). ENCLUS [40] is a variant of CLIQUE that searches for several clusters instead of dense units.

SUBCLU [41] relies on DBSCAN [29] to find clusters in each subspace in a bottom-up manner. From a general point of view, the global density threshold used by grid-based algorithms or algorithms such as SUBCLU however induces a dimensional bias: a low threshold that performs well in low-dimensional space will induce error in subspaces of higher dimension and vice-versa. In order to cope with this problem, Dimensionality UnbiaSed Cluster model (DUSC) [42] has been later introduced. In [43], Liao et al. expose a grid-based clustering algorithm that copes with density variability in feature space which can lead to nested clusters. Their algorithm uses an Adaptive Mesh Refinement (AMR) technique that first applies a grid clustering

[3] A medoid is a value that represents the elements of a cluster or a subset of a set. Its value is always one of the original set.

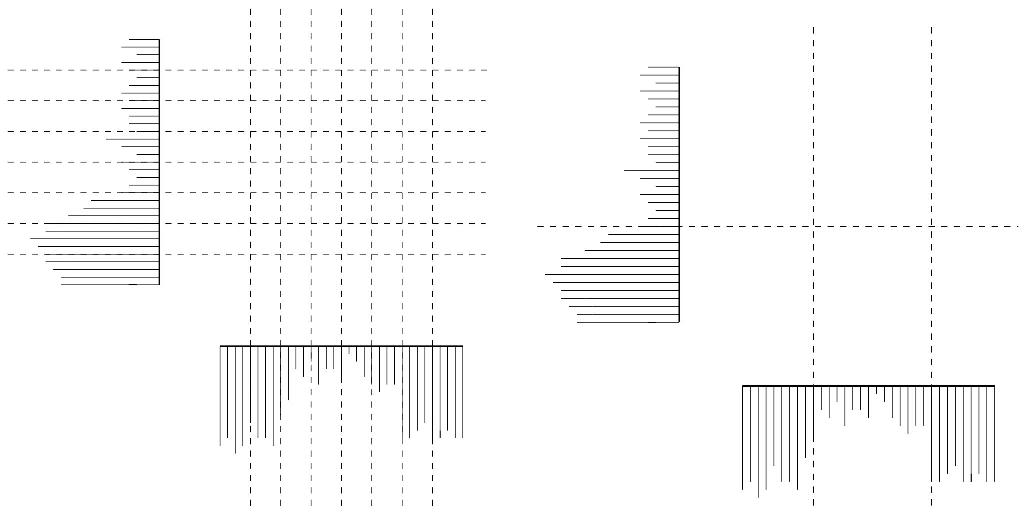

(a) Fixed grid (b) Adaptive grid

Figure 2.2: Examples of grids.

method to the whole feature space. It then builds finer grids onto dense regions in order to find nested clusters. Each node in the produced hierarchical tree is a spatial subdomain of its father. Each node child represents a cluster located inside the considered node. This algorithm is useful for highly irregular datasets. Figure 2.3 displays an example of several nested clusters and their associated AMR tree containing several grids. In this particular case, a first grid, here called “grid 0”, is applied over the whole the feature space. This grid contains two clusters. The first one is subdivided again into “grid 1”. This particular grid does not contain any clusters. The second cluster found in “grid 0” is then analyzed through “grid 2”. In this grid, the algorithm finds two clusters that are analyzed through “grid 3” and “grid 4”. These two grids however do not contain any other clusters. The tree on the right part of Figure 2.3 displays the cluster “genealogy” or the cluster inclusions obtained from the algorithm.

In [44], Akodjenou-Jeannin et al. propose a flexible grid method similar to k-means in the sense that it needs to know the number k of cluster to find. Starting on a single cell containing the whole feature space, it successively builds random hyperplanes in the successive most dense cell in order to create two cells. This process stops when M non-empty cells have been created. Their algorithm then builds a weighted graph. Each edge weight represents the similarity between two cells. This similarity takes into account both density and spatial similarity. The final partition is built by removing the lowest weighted edges until k clusters remain.

The fourth and last subcategory is hybrid clustering. Hybrid clustering algorithms do not aim at “finding an unique assignment of points to subspace clusters” neither at finding all clusters in all subspaces. Density-based Optimal projective Clustering (DOC) [45], MineClus [46, 47], Detecting Subspace cluster Hierarchies (DiSH) [48], Hierarchical approach with Automatic Relevant dimension selection for Projected clustering (HARP) [49], Support and Chernoff-Hoeffding bound-based Interesting Subspace Miner (SCHISM) [50], FIlter REfinement Subspace clustering (FIRES)

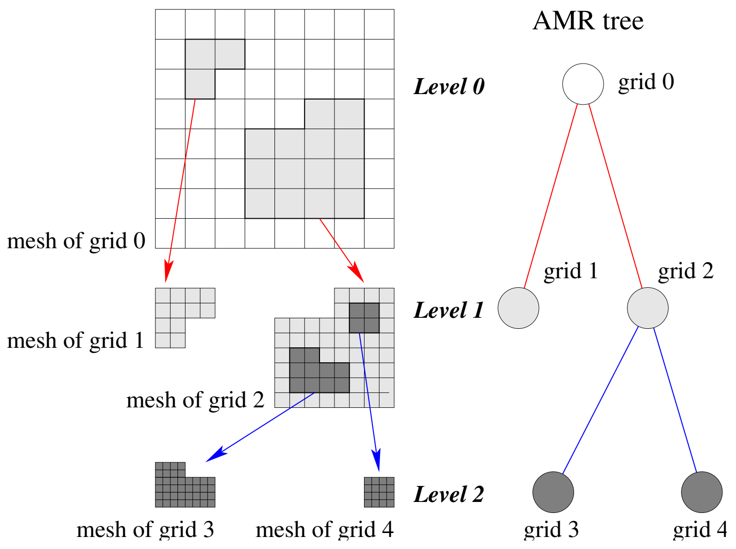


Figure 2.3: Examples of AMR.

[51] and Projected Clustering via Cluster Cores (P3C) [52, 53] can all be classified as hybrid clustering algorithms. DOC uses a global density threshold defined by the side-length w of standard hypercube and the number of point α located in the hypercube. It also uses a third parameter β which specifies the balance between the number of points and the dimensionality of a cluster. It is randomly initialized. Each of its run may find a single cluster. If one wants to find k clusters, DOC has to be run at least k times. MineClus follows a similar idea but proposes a deterministic method to find an optimal projected cluster, given a sample point seed. Mineclus addresses the problem as a frequent item set mining through a modified frequent pattern tree growth. DiSH uses an approach close to PreDeCon in terms of attribute weighting. It however uses a hierarchical clustering algorithm that is able to find clusters with multiple inclusions. HARP is a single-link-like clustering algorithm that uses a modified distance and does not produce a hierarchy of subspace cluster. It successively merges points or clusters provided that the resulting cluster displays a minimum number of relevant attributes. A relevance index that takes into account the cluster size is thus introduced. SCHISM aims at finding interesting subspaces instead of pertinent clusters. In order to achieve this goal, it uses grid-like discretization of the feature-space and a depth-first search with backtracking to find pertinent subspaces. FIRES first computes cluster search in single dimensions. It then defines the similarity of clusters as the number of intersecting points. A global partition can then be obtained by clustering these values. P3C computes one-dimensional intervals that are likely to approximate higher-dimensional subspace clusters. These intervals are then merged through an APRIORI-like bottom-up search strategy. The search result is then refined and the final output is a matrix that specifies the probability of membership of point for each cluster. P3C thus allows both single and multiple clusters membership of points.

Correlation clustering The second category defined in [30] is correlation clustering.

The rationale of this category of clustering algorithms is that clusters can be found inside subspaces where correlations between original attributes are removed. Such subspaces are thus built on arbitrary-oriented axis. Many of the algorithms belonging to this category use PCA. ORCLUS (arbitrarily ORiented projected CLUSter generation) [54] and 4C (Computing Correlation Connected Clusters) [55] both use PCA combined with other clustering algorithms (respectively k-means and density-based) to tackle the considered problem. HiCO (HIerachical Correlation Clustering) [56] uses a hierarchical approach according to a distance based on correlation dimensionality and subspace orientation. A final hierarchy of correlation clusters is built through hierarchical density-based clustering. Another approach is adopted by the algorithm CASH (Clustering in Arbitrary Sub-spaces based on the Hough transform) [57]. Other mathematical tools have been used such as self-similarity [58, 59, 60] and sampling [61, 62].

Pattern-based clustering The third category according to the classification of Kriegel et al. is pattern-based clustering also called biclustering. A common representation for the data in a clustering problem is a matrix where rows represent objects and columns attributes. Pattern-based clustering algorithms seek clusters of instances/rows *or/and* attributes/columns while classical approaches like clustering in axis-parallel subspaces and correlation clustering only attempt to group rows. The term biclustering comes from the search of both similar row and similar columns. Four subcategories of clusters seeked by pattern-based clustering or biclustering are identified: constant biclusters, biclusters with constant rows or columns, biclusters with coherent values and biclusters with coherent evolutions.

Constant biclusters are clusters where several objects have the same values for each of them and for several attributes. The pioneering work for the search of this type of clusters has been accomplished by Hartigan et al. in [63].

Biclusters with constant rows are constituted of several objects exhibiting a constant value for several attributes (this value being different between the objects). Biclusters with constant columns have the same value for one or several attributes across several objects (the values may differ between attributes). In [64, 65, 66], the authors tackle the search of this particular type of clusters.

Biclusters with coherent values exhibit a particular covariance between rows and columns. The first work in this field has been realized by Cheng and Church [67]. The FLOC algorithm, which stands for FLexible Overlapped Clustering [68] and CoClus [69] algorithm tries to find biclusters of this type. FLOC first chose several seed-clusters that are further optimized by randomly removing or adding a row or a column. CoClus can be described as a k-means-like biclustering algorithm.

Finally, biclusters with coherent evolution are clusters exhibiting correlation in the way values change among all objects. In [70, 71], the authors try to find this type of biclusters.

Table 2.1 summarizes this clustering algorithm classification and details each of the four categories presented and their respective subcategories.

Categories	Subcategories	Algorithms
Axis parallel clustering	Projected clustering	PROCLUS [32], FINDIT [33], SSPC [34] and PreDeCon [35]
	Soft-projected clustering	LAC [36] and COSA [37]
	Subspace clustering	CLIQUE [38], MAFIA [39], ENCLUS [40] and SUBCLU [41]
	Hybrid clustering	DOC [45], MineClus [46, 47], DiSH [48], HARP [49], SCHISM [50], FIRES [51] and P3C [52, 53]
Correlation clustering	PCA-based	ORCLUS [54], 4C [55]
	Other	HiCO [56], CASH [57]
Pattern-based clustering	Constant biclusters	Block clustering [63]
	Biclusters with constant rows or columns	[64, 65, 66]
	Biclusters with coherent values	δ-bicluster [67], FLOC [68] and CoClus [69]
	Biclusters with coherent evolution	OP-Cluster [71]

Table 2.1: Clustering algorithm classification

2.1.3.3 Clustering evaluations

Many attempts have been conducted in order to compare and evaluate clustering algorithms. In [30], Kriegel et al. group and classify many clustering techniques. The global structure of our section 2.1.3.2 comes from this work. Clustering algorithm evaluation can however be complicated to realize, especially over a single reference dataset. The main problem being that some of these algorithms target different problems and therefore, cannot be compared (e.g. bi-clustering and axis-parallel clustering).

Several evaluations of clustering algorithms exist in the literature. In [31], Parsons et al. compare several clustering approaches and classify them according to their nature: either top-down or bottom-up. They present CLIQUE, ENCLUS, MAFIA, CBF [72], CLTREE [73], DOC [45], PROCLUS, ORCLUS, FINDIT, δ-Clusters [68] and COSA and evaluate MAFIA and FINDIT. They use a different taxonomy than [30]: they do not separate axis-parallel and arbitrary-oriented clustering algorithms, they instead consider that "Subspace clustering algorithms localize the search for relevant dimensions allowing them to find clusters that exist in multiple, possibly overlapping subspaces". MAFIA exhibits better overall performance than FINDIT. In [74], Muller et al. evaluate several clustering algorithms. They present and test the following algorithms: CLIQUE, DOC, MineClus, SCHISM, SUBCLU, FIRES, INSCY [75], PROCLUS, P3C and STATPC [76]. In [77], Moise et al. analyze and evaluate several axis-parallel clustering algorithms. Following the taxonomy of [30], studied subspace clustering algorithms are MAFIA and DiSH, projected clustering algorithms are PROCLUS, SSPC, HARP, MineClus, P3C, PreDeCon and FIRES. They also include STATPC "which propose a statistical-based approach to projected and subspace clustering". They include several full-dimensional clustering algorithms: k-means [28], EM [78], BAHC [79], DIANA [80], CLARANS [81] and DBSCAN.

Table 2.2 presents each clustering algorithm evaluation work quoted above.

Title	Authors	Reference	Presented algorithms	Evaluated algorithms
Evaluating Subspace Clustering Algorithms	Lance Parson, Ehtesham Haque, Huan Liu	[31]	CLIQUE, ENCLUS, MAFIA, CBF, CLTREE, DOC, PROCLUS, ORCLUS, FINDIT, δ-Clusters, COSA	MAFIA FINDIT
Evaluating clustering in subspace projections of high dimensional data	Emmanuel Müller, Stephan Günnemann, Ira Assent, Thomas Seidl	[74]	CLIQUE, DOC, MineClus, SCHISM, SUBCLU, FIRES, INSCY, PROCLUS, P3C, STATPC	All
Subspace and projected clustering: experimental evaluation and analysis	Gabriela Moise, Arthur Zimek, Peer Kröger, Hans-Peter Kriegel, Jörg Sander	[77]	k-means, DBSCAN, BAHC, DIANA, CLARANS, EM, MAFIA, DiSH, PROCLUS, SSPC, HARP, MineClus, P3C, PreDeCon, FIRES, STATPC	All

Table 2.2: Clustering evaluations.

2.2 Network anomaly detection and intrusion detection

The work presented here reflects advances in designing network anomalies and intrusion detection techniques. Approaches targeting the correlation or combination of network anomaly detection systems such as [82], [83] or [84] are not presented here because they intend to reuse results of network anomaly detection techniques instead of directly monitoring traffic measurements (either packet-based or flow-based). As such, their problematic is located at a higher abstraction level.

2.2.1 Classic network anomaly and intrusion detection

This section addresses the two main classical approaches used by commercial and academic anomaly detection systems: misuse detection based on signatures and anomaly-based detection.

A little reminder of notions addressed in chapter 1.1.2 is needed here. Anomaly-based detection relies on the detection of a deviation from a normal state or profile. Network anomaly detection represents the detection of network anomalies. The use of the "anomaly detection" words in both terms introduces here confusion. These two terms shall not be confused.

2.2.1.1 Misuse detection-based intrusion and network anomaly detection

Misuse detection-based systems detect anomalous events through comparison of occurring events with a sets of anomaly signatures. The two most famous IDS or IPS relying on misuse detection are Snort and Bro [85]. Snort relies on packet inspection to detect intrusions. Snort provides an up-to-date, well-documented, and tested set of rules or signatures. Bro performs a state-full inspection of traffic in order to rebuild 5-tuple flows (source and destination IP addresses, source and destination port numbers and protocol) and to find intrusions among these flows. They both use signatures to detect and signal events to network operators. URCA [86] aims at finding the root cause of an anomaly. It can be plugged to any network anomaly detection algorithm. It then uses 10 features: source and destination Internet Protocol (IP) addresses, source and destination port numbers, input and output router interfaces, previous and next-hop AS numbers, source and destination AS numbers. Identification of anomalous flows is realized by finding flows sharing feature values that impact the used anomaly detection algorithm. Once anomalous flows are identified, the algorithm builds several new features or coordinates. These features are the average flow size in terms of packets, the average packet size in number of bytes, for each of the 10 flow features, the entropy of the distribution of packets per feature value and for each of the 10 flow features, the fraction of feature values in the full link traffic that also appears in the root cause traffic. URCA then uses the feature space constituted by all these coordinates to project both known anomalies and anomalies to classify. It then uses hierarchical clustering to group unclassified anomalies with known ones.

2.2.1.2 Anomaly detection-based network anomaly detection

In order to cope with unknown anomalies, researchers have designed anomaly detection-based systems relying on the search of a deviation from a normal state.

In [87], Zhang et al present a unified framework for anomaly detection. They propose to separate anomaly detection in two categories: systems using either temporal correlation (among link or nodes) or spatial correlation in order to identify normal traffic. We will follow the same classification.

- We subdivide temporal correlation methods into three families: methods relying on forecasting, techniques relying on signal processing and systems using inner characteristics of traffic and associated hypotheses. The first family of temporal correlation methods associates anomalies with deviation from forecasted behaviors. The standard techniques include the following models: Auto-Regressive Integrated Moving Average or ARIMA models and deltoids [88]. An ARIMA model is noted ARIMA(p,d,q). The values p, d and q are the autoregressive, integrated, and moving average orders of the three corresponding terms of ARIMA models. Famous ARIMA models include Exponentially Weighted Moving Average, EWMA, which actually is ARIMA(0,1,1) and linear exponential smoothing or Holt-Winters which is equivalent to ARIMA(0,2,2). The seminal work in this family is [17] which applies Holt-Winters forecasting to byte counts. In [89], the authors apply several ARIMA models on traffic-based sketches to detect anomalies. In their work, the use of sketches is motivated by the need to avoid per-flow statistics and only use ARIMA models on a few subspaces/sketches. In [90], Roughan et al. introduce a correlation mechanism to use both SNMP and Border Gateway Protocol (BGP) time-series in order to find anomalies. They respectively apply Holt-Winters and EWMA over these data. In [91], Soule et al. use Kalman filters to extract normal traffic from the traffic matrix. They then use several other tools such as variance, Cummulative Summation and Generalized Likelihood Ratio test, multi-scale analysis using variance and multi-scale variance shift. In [88], Cormode et al. introduce novel algorithms to find significant changes in network stream data. They introduce the concepts of deltoids which can measure absolute differences, relative differences and variational differences inside network metrics time-series. In [92], the authors use a non-Gaussian multi-resolution models applied to sketches in order to detect anomalies on single-link traffic. Their correlation technique is here not spatial but among sketches. This multi-resolution model detects changes in the short-time correlation structure instead of volume changes detected by change-based anomaly detection system. In [93], Brauckhoff et al. associate anomalies with a variation on flow features distributions in two successive time bins. The variation is measured through the Kullback-Leibler divergence. The used flow features are source IP address, destination IP address, source port number, destination port number and flow size in packet. Once anomalies are detected, they are characterized with association rule mining [94].

The second family of techniques in the temporal correlation category are relying on signal processing. Barford et al. in [95] use wavelet filters to detect anomalies on single-link network traffic. They first split traffic into three components (low, medium and high frequency bands) and then find anomalies in medium and high frequency bands by applying a threshold over the local deviation of variance. A great number of work have also tried to use PCA [25, 24] in order to extract anomalous traffic. As explained in section 2.1.1, PCA is an unsupervised learning technique that aims at finding correlated signals. It uses linear combination of original axis of a space to generate new axis in which each uncorrelated signals (called principal components) are successively projected by order of decreasing variance. The characteristics of normal traffic can be captured by a fixed number of components built by PCA. Anomalous traffic is then supposed to be located outside of these components. Methods relying on several measurements made from different points use spatial correlation to extract normal traffic. These methods are presented in the next item. Techniques presented here use mathematical tools to create different point of views from traffic captured on a single measurement point. They thus use the correlation between these “artificial” points of view to extract normal traffic. In [96], Brauckhoff et al. use a Karhunen-Loeve expansion to improve PCA in order to take into account temporal correlations on single-link traffic. In [97], Kanda et al. propose a PCA-based detector designed to work on single-link backbone traffic. The used metric is packet counts. Their method has 4 stages. First, they build N subparts of traffic through sketches using source IP addresses. Secondly, they use sketches to build a second levels of N' traffic subparts out of each of the N traffic subparts. Thirdly, they apply PCA on N groups of N' traffic subparts. Fourthly, they identify anomalous source IP addresses by taking the intersections of anomalous source IP addresses sets. They compare their algorithm with the one of Dewaele et al. in [92] using ground truth provided by heuristics. They show that their algorithm finds different anomalies from the ones found by Dewaele et al. They also introduce an heuristic to automatically determine the number of first components of PCA. They chose to use the number of component that allows them to capture 70% of the cumulative proportion of the first components. The 70% threshold is empirically chosen.

In [87], Zhang et al. present and compare several methods from these two first families: EWMA, Holt-Winters, Fourier analysis through Fast Fourier Transform (FFT), Wavelet analysis as in [98] and temporal PCA. They find that ARIMA and Sparsity-L1 are the best methods to respectively forecast traffic volumes and inferences Origin-Destination flows (OD-flows) from traffic matrix.

The third family contains methods which detect anomalies by making hypothesis over traffic flows nature. ASTUTE (A Short-Timescale Uncorrelated-Traffic Equilibrium) presented in [99] relies on equilibrium properties over traffic flows. In the ASTUTE model, flows crossing a link of interest are generated by a discrete-time marked point process [100], where the mark process determines both the flow's duration and its volume per time bin. Considered flows are standard 5-tuple flows. The used equilibrium properties are the following: independence between flows characteristics (start time, duration and volumes) and stationarity over time regarding the distribution of the flow arrival process and the mark process. The former property is backed by Barakat et al. in [101] and Hohn et al. in [102] while the latter is experimentally verified on real traffic. ASTUTE is thus able to identify anomaly that methods such as Kalman filters or Wavelet are unable to find.

- The other category of anomaly detection system relies on the spatial correlation of traffic. The pioneering contribution in this category are the papers published by Lakhina, Crovella et Diot in [18, 103, 104] on network-wide traffic anomaly detection. The rationale of this technique is that while anomalies only affect few links, normal traffic has the same characteristics across links. Thus, by applying PCA to the whole traffic matrix, the normal traffic characteristics are captured by the first components extracted by PCA. The number of the first principal components used that represent normal traffic is determined through careful traffic analysis. The metrics or features successively used in these papers are: byte counts in [104], packets counts, byte counts and IP flow counts in [103] and entropy values for distributions of several features (source IP address, destination IP address, source port, and destination port) in [18]. In [104], the authors show that this technique is able to detect anomalies, identify anomalous OD-flows and quantify them in terms of byte counts. In [18], they introduce an anomaly classification technique relying on clustering that enables them to group similar anomalous OD-flows. In [105], Li et al. extend this work by adding a sketch-based processing step that is capable to identify anomalous IP flows. In [106], Chhabra et al. use a spatial correlation method to process byte counts at two levels: first, between links on a router, and second, between adjacent routers. This avoids the use of a central processing where the traffic matrix is processed. Anomalies are detected by the generalized quantile sets and false discovery rate, and are associated with outliers [4].

  However, PCA exhibits several limitations. For example, "normal" principal components can be poisoned and thus allow anomalies to stay undetected [7, 107, 108].

[4]An outlier belonging to a point set is a point isolated from the majority of the points of this point set.

Moreover, in [7], Ringberg et al. expose three other weaknesses of PCA-based anomaly detection. First, the false-positive rate [5] is very sensitive regarding the number of normal components. The lack of reliable heuristics to devise this number of components increases the overall lack of robustness. Second, they state that PCA effectiveness is very sensitive to the used traffic aggregation: input links, OD-flows or ingress routers. Finally, they prove that anomalous flow identification relying on PCA only is difficult because of the very nature of PCA. In [109], Rubinstein et al. introduce ANTIDOTE in order to improve poisoning resilience. It uses a modified version of the PCA-subspace method called PCA-Grid. Another attempt toward the same goal is presented in [110]. Abdelkefi et al. use Principal Component Pursuit and show that it is more robust than the classic PCA and the extension presented in [96].

PCA is an unsupervised method per se (cf. section 2.1). In the case of anomaly detection, and as stated in [7], the number of principal component that represents traffic is a critical parameter. In [18, 103, 104], Lakhina et al. use an empirically determined number of principal components. In [97], Kanda et al. use cumulative variance captured by the first principal components as a criterion to find this number. In the light of the sensitivity of PCA towards the number of principal components, an unsupervised use of PCA would require heuristics (such as the one used in [97]) to automatically determine the number of principal components. However, Ringberg et al. quote the cumulative variance captured by the first principal components as a criterion to avoid. According to them, this heuristic shall not be used because different networks exhibits different disparity. This result is consistent with the conclusions of [97]. In fact, Kanda et al. used this criterion on the MAWI dataset only. Cumulative variance may therefore be used to determine the number of principal components when used traffic is consistent. This allows one to keep the same threshold on the cumulative variance and thus use PCA in an unsupervised way. We therefore choose to classify PCA-based systems as anomaly-based network anomaly detection systems.

2.2.2 Machine learning applied to network anomaly detection and intrusion detection

Work in the anomaly detection and intrusion detection fields have been trying to apply machine learning in order to improve performance of classical approaches. The first of the next three sections covers supervised network anomaly detection. Compared to network anomaly detection, the intrusion field has had a much more prolific application of machine learning. They embraced both supervised and unsupervised learning in order to build intrusion detection algorithms.

Several papers provides comparison between such algorithms. In [111], Sadoddin et al. provides a good introduction to machine learning applied to intrusion detection and compares several algorithms (K-means, C-means, Expectation Maximization (EM), Self-

[5]The false positive rate is a statistical measure of the performance of a binary classification test. In the network anomaly detection case, it is the ratio of the number of normal instances falsely classified as anomalous over the total number of normal instances.

Organizing Maps (SOM), Y-means, Improved Competitive Learning Network (ICLN) [112]. This work also introduces the two possible modes of clustering techniques, indirect and direct modes, which respectively are supervised mode and unsupervised mode. In [113], Gogoi et al. survey outlier detection techniques applied to anomaly detection-based intrusion detection. They address work using both supervised and unsupervised approaches. They present work from different authors, among them [114], that we also present later.

It is interesting to note that every intrusion detection technique presented here is evaluated on the KDD dataset [115] (at the notable exception of [116] which targets Address Resolution Protocol (ARP) traffic). The KDD dataset contains both network and system call traces with ground truth labeled anomalies and intrusions. It thus allows one to use both supervised and unsupervised techniques. The availability of such dataset may explain the wide use of machine learning in the intrusion detection field. It also provides a possible cause for the emergence of unsupervised techniques in the intrusion detection field.

The next sections successively address supervised learning-based network anomaly detection, supervised learning-based intrusion detection and unsupervised learning-based intrusion detection.

2.2.2.1 Misuse and anomaly-based supervised network anomaly detection

Literature contains several examples of work applying machine learning to the network anomaly detection problem. In [19], Shon et al. apply several machine learning techniques to anomaly-based network anomaly detection systems. They use several techniques such as Self-Organizing Feature Map (SOFM), Genetic Algorithms (GA), and Support Vector Machine (SVM) in order to first build a profile of normal traffic and then detect novel anomaly deviating from this profile. In [20], Duffield et al. demonstrate that a misuse detection system using flow features and trained with Snort alarms can be as efficient as Snort itself without the expensive payload inspection. In [117], Zi et al. use a modified version of the k-means clustering algorithm called Modified Global K-Means (MGKM) to characterize DDoS attacks. They first inspect traffic in order to select variables that will be used for the clustering. The selected variables are entropy of source IP address and port number, entropy of destination IP address and port number, entropy of packet type, number of packets, occurrence rate of packet type (Internet Control Message Protocol (ICMP), User Datagram Protocol (UDP), Transmission Control Protocol (TCP), SYN). They then apply the MGKM algorithm over the KDD data set. Feature ranking is applied on the clustering results knowing the expected result of the clustering algorithm. The result from this step are then used to recalculate the clusters in a more efficient manner. The system is thus able to find an optimal set of features that characterizes the targeted anomaly (here DDoS).

The use of machine learning for both subcategories of network anomaly detection (misuse and anomaly detection) proves that machine learning can be very useful in order to automate the tuning of the previously exposed algorithm. However, as stated in 1.1.2, they all rely on the availability of training data for either normal or anomalous traffic which in both cases is difficult to obtain.

2.2.2.2 Supervised intrusion detection

Many contributions in the intrusion detection field use supervised learning in order to characterize both normal and anomalous traffics. The majority of work use a two steps technique to proceed to supervised learning. They first train their algorithm in order to build normal and anomalous profiles. Then, they apply the built profile to the dataset to test. In [118], Kuang et al. apply an algorithm called Combined Strangeness and Isolation measure K-Nearest Neighbor (CSI-KNN) on normal and anomalous training sets. In [119, 120], the authors present a method called CCA-S (Clustering and Classification Algorithm - Supervised) to incrementally build models of anomalous and normal traffic. In [121], Burbeck et al. use a customized version of the clustering algorithm BIRCH [122] to first built a model of normality and then incrementally update this model in order to cope with normality changes. In [116], the authors train their algorithm composed of k-means and ID-3 tree over normal ARP traffic and then identify anomalous behavior The Audit Data Analysis and Mining (ADAM) presented in [123] applies a slightly different method over the KDD dataset. Instead of using a two steps method relying on training and testing on normal or anomalous labeled datasets, they use a three steps method: they first train their normal models on anomaly-free data, then, they build anomalous models by using the normal model as a negative over real data, and finally they test their system on real traffic.

2.2.2.3 Unsupervised intrusion detection

The other axis of machine learning use for intrusion detection is their direct application in the sense of [111], i.e. through unsupervised learning. The seminal work in this field are the articles by Portnoy et al. [23] and Eskin et al. [124]. In [23], the authors build a system that uses the unlabeled training data from KDD in order to train their system. The anomalousness of each data segment is assessed by the hypothesis 1.2. This is the first fundamental advance towards unsupervised training-independent IDS: it allows their system to avoid the use of expensive-to-produce labels in order to work. They then improve this first advance with a new system [124] that does not require training at all. In [124], they compare three different algorithms, cluster based estimation, K-Nearest Neighbor and one class SVM, to build partitions over test data on-the-fly.

In [125, 126], the authors also attempt to apply unsupervised technique to intrusion detection. In [126], Oldmeadow et al. adapt fixed-width clustering to time-varying traffic characteristics in order to cope with traffic variability. They also introduce a feature weighting mechanisms based on manual inspection. In [125], Leung et al. modify the pMAFIA [39] clustering algorithm (previously presented in section 2.1.3.2) to create fpMAFIA (frequent pattern in pMAFIA) algorithm. They thus provides a greater scalability at the expenses of accuracy.

In [114], Zhang et al. use outlier detection to find intrusion inside the KDD dataset. They add an outlier detection capacity to the random forest clustering algorithm.

2.3 Summary

Unsupervised learning techniques have been created and designed for a very long time. They aim at extracting information from data with using little knowledge about its structure. Three family of such methods exists: blind signal separation, neural network models and clustering. We present each of these methods in this chapter. We put a special emphasis on clustering since it is the the method that we later chose to use. The following chapter further motivates this choice.

We also presented a non exhaustive picture of previously proposed network anomaly detection algorithms. We classify these systems into two categories: temporal correlation-based and spatial correlation-based. We also list several work that apply machine learning techniques in the network anomaly detection field but also in the intrusion detection field. Machine learning applications in the intrusion detection field are interesting in the sense that contrary to network anomaly detection, they use unsupervised techniques.

The following chapter presents our own application of unsupervised learning to the network anomaly detection field.

Chapter 3

Unsupervised network anomaly detection

This chapter addresses the design of our network anomaly detection algorithm. We named it UNADA which stands for Unsupervised Network Anomaly Detection Algorithm. As previously exposed in sections 1.1.3 and 1.2, it aims at finding or mining anomalies inside traffic without using any previous knowledge about its structure. In order to accomplish this task, it relies on unsupervised learning and especially clustering and on the two simple hypothesis presented in section 1.1.3.

The algorithm runs in three consecutive stages. Firstly, a pre-processing step is applied on network traffic. During this step, a time series based change detection algorithm is applied on several simple metrics such as number of bytes, packets or flows. Any time slot flagged as anomalous is then processed through multi-resolution flow aggregation and feature computation to build traffic features on aggregated flows. This first step is depicted on the upper part of Figure 3.1 and inside the dotted frame with indice 1. The unsupervised learning based detection algorithm is depicted on the lower part of Figure 3.1 and is located inside the dotted frame with indice 2. This step begins in the second stage, using as input the set of flows built by the first step. Concepts presented in this section have been published or accepted in [127, 128, 129, 130, 131] and exposed in ECODE deliverables D3.2 [132] and D3.3 [133]. The third step process anomalous flows extracted by the second step and is detailed in the next chapter. This step is contained inside the dotted frame with indice 2 on Figure 3.1.

- Section 3.1 describes change detection, flow aggregation and attribute computation (part 1 in Figure 3.1).
- Section 3.2 exposes our unsupervised algorithm to detect anomalies (part 2 in Figure 3.1).
- Chapter 3.3 presents an evaluation of the computational time of our algorithm.

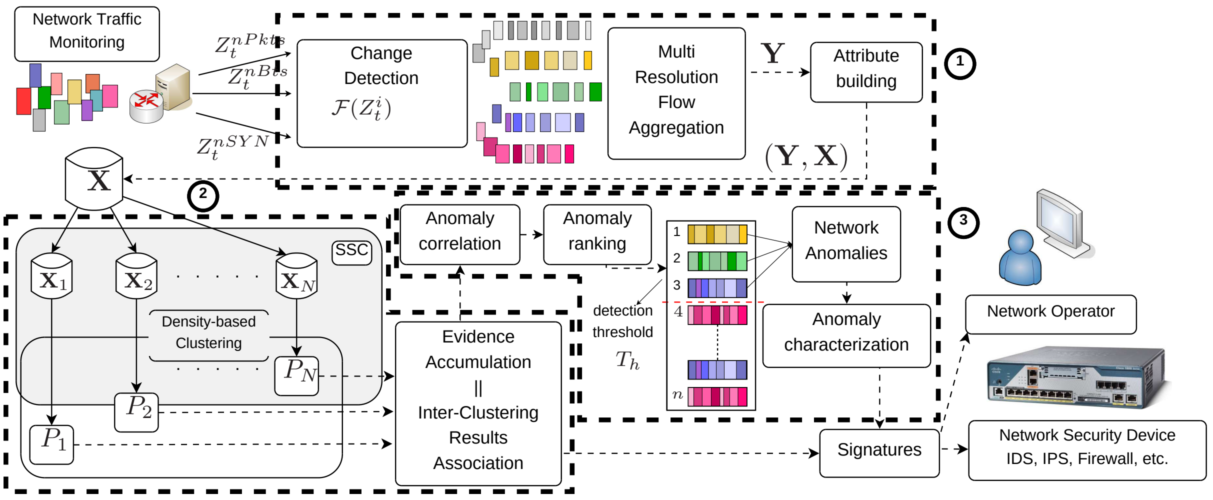


Figure 3.1: High-level description of our approach.

3.1 Change-Detection, Multi-resolution Flow Aggregation & Attribute computation

This first stage of UNADA aims at finding evidence of an occurring anomaly and in this case, to process and format traffic in order to provide semantically interesting data for the unsupervised detection step. The techniques described in this section are represented on Figure 3.2.

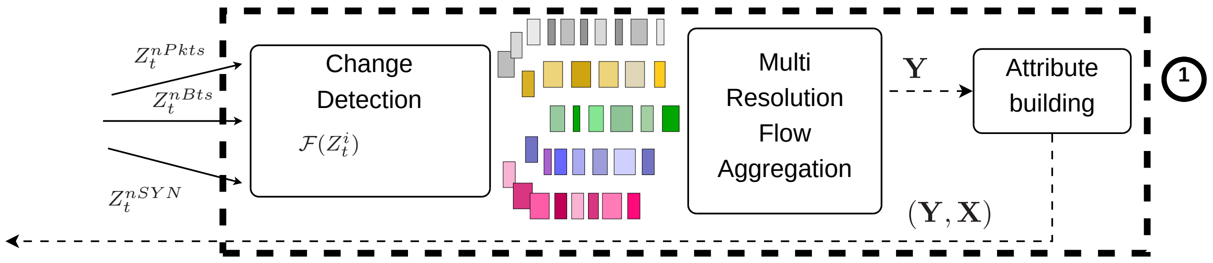


Figure 3.2: High-level description of our approach.

UNADA is designed to process single-link packet-level traffic. Our algorithm works on traffic captured in consecutive time slots of fixed length, noted ΔT or ws. UNADA first takes as input the whole traffic and tries to find evidences that an anomaly may be occurring. In case of a detected anomaly, traffic is aggregated into flows through several aggregation levels. Finally, several *features* or *attributes* are built in order to provide insights on flow nature. The next three sections present these three steps.

3.1.1 Change detection

The first step of UNADA is the change detection step. To detect an anomalous time slot, time-series Z^i are constructed for simple traffic metrics such as number of bytes, packets, and IP flows per time slot. Any generic change-detection algorithm $\mathcal{F}(.)$ based on time-series analysis (cf. section 2.2.1.2) is then used on Z^i, Z^i being the time-series for the metric of index i and Z_t^i being the value of the metric i for the t^{th} time slot. At each new time slot, $\mathcal{F}(.)$ analyzes the different time-series associated with each metric. time slot t_0 is flagged as anomalous if $\mathcal{F}(Z_{t_0})$ triggers an alarm for any of the traffic metrics.

In our implementation, the change-detection algorithm $\mathcal{F}(.)$ used deltoids as they are defined in [88].

3.1.2 Multi-resolution Flow Aggregation

Once an alarm has been raised by the change detection stage, multi-resolution flow aggregation is applied on the traffic. IP flows can be aggregated at different *flow-resolution* levels. The flexibility of our multi-resolution aggregation system allows us to use many aggregation levels (like *source Network Prefixes* ($l_{1,2,3}$: IPsrc/8, /16, /24) and *destination Network Prefixes* ($l_{4,5,6}$: IPdst/8, /16, /24) if needed.

3.1.3 Attribute building

The attribute computation step is depicted on the right of Figure 3.2. It takes as input all packets and aggregated flows in the flagged time slot. Its goal is to generate meaningful attributes that will further allow us to separate normal traffic from anomalous one.

We propose a framework to generate attribute over traffic. Its design aims at providing modularity and easy modification.

The remaining of the section is structured as follows: we first define a *metric*, then, we explain how metrics are built and finally, we describe the technique used to create attribute over metrics.

3.1.3.1 Metric definition

We define two types of metrics over a set of packets. The first type of metrics is called simple metric. It can only contain a single value. This type of metric can, for example, be used to store the total number of packet in the set. The second type is a distribution metric, it contains a distribution i.e., a structure that stores several values and their respective number of occurrences. Distribution metric can be used to store the source IP addresses.

3.1.3.2 Metric building

We identify two types of data inside network packets: meta-data and data contained in packets. Meta-data is data that represents packets characteristics that are not embedded in the payload, i.e.: number of packets in the set, size of the packet etc. Meta-data is described through a simple metric. Table 3.1 gives two examples of simple metrics related to metadata.

Metric name	Type of metric
nb_packets	Simple metric
size_of_packets	Simple metric

Table 3.1: Metrics built on meta-data.

The second type is the actual data contained in the payload of the packet. Inside the payload, we separate each field of a packet header in two categories: first, fields that can only take two values ($n = 2$), which usually mean true or false. This type of field is represented as a simple metric. This simple metric counts the number of packets in the set that have the specific field set to true. The second type of field can have a number of values $n > 2$. In this case, each field is represented by a distribution metric. Table 3.2 presents the metrics used to characterize the packet payload.

Metric name	**Type of field in header**	**Type of metric**
src_addr	Int32	Distribution metric
dest_addr	Int32	Distribution metric
src_port	Int	Distribution metric
dest_port	Int	Distribution metric
nb_icmp_packet	Bit	Simple metric
nb_echorequestreply_packet	Bit	Simple metric
nb_syn_packet	Bit	Simple metric
nb_rst_packet	Bit	Simple metric

Table 3.2: Packet fields and their associated metrics used by our algorithm.

3.1.3.3 Attributes generation over packet sets

Once metrics are built over aggregated flows [1], the UNADA algorithm builds attributes on top of them. Each distribution (simple or distribution) generate *intermediate attribute(s)* that are further used to build *attributes*.

A simple metric can be transformed into a single *intermediate attribute* called *value* which contains the value of the simple metric. A distribution metric generates two *intermediate attributes*. The first one is *nb_of_elements*. It represents the number of different values in the distribution. The second intermediate attribute generated is *ratio_bigst_occur_over_nb_occur*. As its name states, it represents the ratio between the biggest occurrence in the distribution and the total number of occurrences. This intermediate attribute can be related to entropy: a high entropy will give a ratio value close to 0 while a low entropy is equivalent to a value close to 1.

Used attributes are listed in table 3.3. The attribute *nb_destinations* is built from the intermediate attribute *nb_of_elements* of the distribution metric *dest_addr*. The same goes for every other attributes with different simple or distribution metrics. The attributes presented in table 3.3 are the ones used throughout this manuscript.

It is interesting to notice that when flows are not aggregated, some of these attributes have a very low semantic value. For example, if one considers traffic flows aggregated according to the destination IP address, the attribute *nb_destinations* is of poor interest. One can also note adding a new attribute on a new metric is extremely simple thanks to the modularity and expressiveness of our framework.

[1] In order to reduce the amount of processing, metrics are first built on atomic flows. Metrics of aggregated flows are later obtained by fusionning metrics belonging to flows aggregated together.

Attribute	**Description**	**Intermediate attributes used**	**Metric used**
nDests	# different destination IP addresses	*nb_of_elements*	dest_addr
nSrcs	# different source IP addresses	*nb_of_elements*	src_addr
nPkts/ nDiffDestPort	# packets divided by # different destination port	*nb_of_elements*	nb_packet, dest_port
nDiffSrcAddr/ nDiffDestAddr	# different source IP address divided by # different destination IP address	*nb_of_elements*	src_addr, dest_addr
nICMP/nPkts	# icmp packets divided by # packets	*value*	nb_icmp_packet, nb_packets
nEchoReqReply/ nPkts	# ICMP echo-request or reply packets divided by # packets	*value*	nb_echorequestreply_packet, nb_packets
nSYN/nPkts	# syn packets divided by # packets	*value*	nb_syn_packet, nb_packets
nRST/nPkts	# rst packets divided by # packets	*value*	nb_rst_packet, nb_packets
bgstDestPort/ tNbOccuDestPort	# occurence of most occuring destination port divided by total # of destination port	*ratio_bigst_occur_over_nb_occur*	dest_port

Table 3.3: Attributes of aggregated flows derived from traffic.

3.1.4 Summary

The first step of our algorithm is a pre-processing step which perform several successive tasks. We first try to find anomalous behaviors by looking at traffic volumes time-series. Then in case of suspected anomaly, we use a multi-resolution aggregation scheme to build aggregated traffic flows. Finally, we create several attributes on traffic to characterize the previously built flows.

Once a time slot has been flagged and attributes have been built, UNADA tries to find anomalies inside the traffic. In order to achieve this goal, it uses unsupervised learning techniques.

3.2 Finding anomalies with unsupervised learning

Following change-detection, multi-resolution flow aggregation and attribute generation, our algorithm uses unsupervised learning technique to extract anomalies. Our problem can be formulated like this: considering the hypotheses in 1.1.3, we want to extract normal and anomalous traffic parts. Let $\mathbf{Y} = \{\mathbf{y}_1, .., \mathbf{y}_F\}$ be the set of F aggregated-flows in the flagged slot considering a specific aggregation. We note the number of hosts H. The value of F is thus directly linked to H through $F \leq H$. Each flow $\mathbf{y}_f \in \mathbf{Y}$ is described by a set of A traffic *attributes* or *features* (cf. section 3.1.3.3 and table 3.3). Let $\mathbf{x}_f \in \mathbb{R}^A$ be the corresponding vector of traffic features describing flow $\mathbf{y}_f$, and $\mathbf{X} = \{\mathbf{x}_1, .., \mathbf{x}_F\}$ the complete matrix of features, referred to as the *feature space*. The unsupervised detection starts after the steps presented in section 3.1 and aims at assessing abnormality of each of $\mathbf{y}_f$ flow using unsupervised learning.

Section 3.2.1 addresses how anomalous and normal traffic parts look like in the feature space. Section 3.2.2 explains why we chose to create our own clustering algorithm. Section 3.2.3 details our clustering approach. The unsupervised learning-based algorithm is then further detailed in the two final sections: section 3.2.4 exposes splitting through subspace clustering algorithm and section 3.2.5 addresses combining through Evidence Accumulation and Inter-Clustering Result Association.

3.2.1 Traffic representations in feature space

This section details what are the representations of anomalous and normal traffic parts in the *feature space*. It is organized in two parts: we detail how the normal traffic is represented inside the feature space, and then, we address the anomaly representations.

3.2.1.1 Normal traffic representations in feature space

Section 3.1.2 states that the used traffic representation instances are aggregated flows. The hypothesis 1.2 is hence refined in this context:

Hypothesis 3.1 *The majority of aggregated flows are normal traffic. Which is equivalent to: $Y\%$ of aggregated flows are normal with $Y > 50\%$.*

Hypothesis 3.1 states that the majority of the aggregated flows is normal. The hypothesis 1.1 says that anomalies are statistically different from normal traffic. Considering the whole *feature space*, normal traffic can then be viewed as one or several cluster(s) containing at least half of the aggregated traffic flows. This hypothesis is refined for our experiments in section 3.2.6 and is further discussed in section 6.2.1.

3.2.1.2 Anomalies representations in feature space

This section aims at describing how anomalies are represented inside the feature space in terms of clusters and outliers. Anomaly characteristics and their impact on anomaly representation inside the feature space is detailed in table 3.4. Traffic anomalies can be roughly grouped in three different classes, depending on their spatial structure and number of impacted IP flows: *1-to-1* anomalies, *1-to-N* anomalies and *N-to-1* anomalies. *1-to-1* anomalies are point-to-point anomalies such as DoS or port scan. *1-to-N* anomalies involve many IP flows from the same source towards different destinations; examples include network scans and spreading worms/virus. On the other hand, *N-to-1* anomalies involve IP flows from different sources towards a single destination; examples include DDoS attacks and flash-crowds.

The distributed nature of anomalies have a great impact on their representations inside the feature space. In fact, *1-to-1* or point-to-point anomalies include a single flow and will therefore always constitute a single outlier no matter which aggregation levels l_i is used. Distributed anomalies (i.e. *1-to-N* or *N-to-1*) behave differently. If, for example a network scan targets several machines located in T /24 prefixes which are themselves contained in a single /16 prefix. Then, if one looks at traffic aggregated by IPsrc/8 or /16 or /24, there will be a single flow from the source of the scan. This flow will be represented inside the feature space as an outlier. If one looks at traffic aggregated by IPsrc/8 or /16, he will also observe a single flow targeting all the machines located in the same /16 prefix. This flow will also be an outlier. However, if one aggregates traffic through IPsrc/24, the scan will be constituted by T flows, each targeting one of the T /24 prefixes. The network scan will therefore be represented as a cluster of T points or flows.

Clusters, by definition, contain several points and thus can be considered as a greater evidence of anomalousness than outliers. Using IPsrc key permits to find cluster representing *1-to-N* anomalies and thus highlight them, while *N-to-1* anomalies are more easily detected with IPdst key. The choice of both keys for clustering analysis ensures that even highly distributed anomalies, which may possibly involve a large number of IP flows, can be represented as clusters or outliers.

Anomaly	**Distributed nature**	**Aggregation type**	**Representation in feature space**	**Impact on traffic features**
ICMP DoS	1-to-1	IPsrc/$*$	Outlier	nSrcs = nDsts = 1, nICMP/nPkts $> \lambda_1$,
		IPdst/$*$	Outlier	nEchoReqReply/nPkts $> \lambda_2$
SYN DoS	1-to-1	IPsrc/$*$	Outlier	nSrcs = nDsts = 1, nSYN/nPkts $> \theta_1$,
		IPdst/$*$	Outlier	bgstDestPort/tNbOccurenceDestPort $> \theta_2$
DDoS ICMP to several @IP/24	N-to-1	IPsrc/24 (l_3)	Cluster	nDsts = 1, nSrcs $> \beta_1$,
		IPsrc/16 (l_4)	Outlier	nICMP/nPkts $> \beta_2$.
		IPdst/$*$	Outlier	nEchoReqReply/nPkts $> \beta_3$,
SYN DDoS to several IP/24	N-to-1	IPsrc/24 (l_3)	Cluster	nDsts = 1, nSrcs $> \delta_1$
		IPsrc/16 (l_4)	Outlier	nSYN/nPkts $> \delta_2$.
		IPdst/$*$	Outlier	bgstDestPort/tNbOccurenceDestPort $> \delta_3$
SYN DDoS attack response to several @IP/24	1-to-N	IPsrc/$*$	Outlier	nSrcs = 1, nDests $> \alpha_1$, nRST/nPkts $> \alpha_2$
		IPdst/24 (l_6)	Cluster	
		IPdst/16 (l_7)	Outlier	
Port scan	1-to-1	IPsrc/$*$	Outlier	nSrcs = nDsts = 1,
		IPdst/$*$	Outlier	nSYN/nPkts $> \nu_1$
Network scan to several @IP/24	1-to-N	IPsrc/$*$	Outlier	nSrcs = 1, nDests $> \rho_1$, nSYN/nPkts $> \rho_2$, bgstDestPort/tNbOccurenceDestPort $> \rho_3$
		IPdst/24 (l_6)	Cluster	
		IPdst/16 (l_7)	Outlier	
Spreading worms to several @IP/24	1-to-N	IPsrc/$*$	Outlier	nSrcs = 1, nDsts $> \eta_1$, nSYN/nPkts $> \eta_1$, bgstDestPort/tNbOccurenceDestPort $> \eta_2$
		IPdst/24 (l_6)	Cluster	
		IPdst/16 (l_7)	Outlier	

Table 3.4: Anomalies detailed according to distributed nature, representations in feature space and impact on traffic features. Anomalies of distributed nature 1-to-N or N-to-1 involve several /24 (sources or destinations) addresses contained in a single /16 address.

Table 3.4 describes the impact each anomalies on several traffic attributes or features. All the thresholds used in the description are introduced to better explain the impact of an anomaly over some of these features. DoS/DDoS attacks are characterized by many packets sent from one or more source IPs towards a single destination IP. These attacks generally use particular packets such as TCP SYN or ICMP echo-reply, echo-request, or host-unreachable packets. Port scans involve packets from one source IP to a single destination IP, and are usually performed with SYN packets. Network scans are characterized as flows containing packets from one source IP to a single destination ports towards several destinationIPs, and using SYN packets. Spreading worms differ from network scans because they are directed towards a small specific group of ports for which there is a known vulnerability to exploit (e.g. Blaster on TCP port 135, Slammer on UDP port 1434, Sasser on TCP port 455). Some of these attacks can use other types of traffic, such as FIN, PUSH, URG TCP packets or small UDP datagrams.

3.2.2 Why we create our own clustering recipe ?

The unsupervised learning technique that we will use need to be able to:

- Detect both outliers and clusters in the feature space.
- Use the original easy-to-understand attributes in order to ease the work of network operators.

According to our state of the art in section 2.1, three techniques are available. Techniques of the Blind Signal Separation extract hidden signals in several mixed relying on different hypotheses like correlation (PCA) or independence (ICA). Such techniques use linear combinations between dimensions or attributes to extract correlated or uncorrelated signals. Since we want to keep easy-to-understand attributes, we shall avoid this family of techniques that use linear combinations of the original features. Techniques belonging to the neural network family like self-organizing maps do not have a clear and established corpus of methods to extract clusters and outliers from feature space despite work targeting this goal have been published like in [26]. On the other hand, clustering techniques explicitly try to group similar instances and thus fit perfectly our needs.

Among the clustering techniques exposed in 2.1, axis-parallel clustering methods such as PROCLUS, MineClus or MAFIA are the ones that best suit our constraints since they are designed to build clusters *and* keep original attributes.

Chronologically, we first quickly prototype unsupervised learning in Matlab [2]. We test several classic algorithms like the Matlab built-in k-means and a freely available Matlab implementation of DBSCAN [3]. We established that these techniques are not suited for clustering in high-dimensional data, but work well in low-dimensional spaces. At this point, two options are available to us: either re-use these results to build partitions for the whole feature space using our own algorithm or use any of the techniques exposed in section 2.1. We chose the first solution.

There are several reasons that explain this choice. First, the vast majority (if not all) of the algorithms previously exposed are implemented inside the ELKI framework [134] or the WEKA framework [135]. These frameworks aim at easing the use of machine learning algorithms for non-specialists by providing simple interfaces to many machine learning techniques. They are both implemented in Java. On the other hand, our tool is implemented in Ocaml [4]. This choice has been motivated by OCaml features: conciseness strong typing, type inference, multi-paradigm programming (functional, imperative and class-oriented) and the fact libpcap [5] bindings already exist. A more complete advocacy for the use of Ocaml can be found in [136]. OCaml to Java bindings are possible through CamlJava [6]. However, the use of Java introduces a potential performance problem.

[2]http://www.mathworks.fr/products/matlab/index.html

[3]https://code.google.com/p/dmfa07/downloads/detail?name=DBSCAN.M

[4]http://caml.inria.fr/

[5]http://www.tcpdump.org/

[6]http://forge.ocamlcore.org/projects/camljava/

Moreover, such algorithm would have appeared to our code as a black box and thus hard to debug (if needed) since we have limited knowledge about their internal implementation. A solution to these problems is to re-implement the empirically determined best fitted clustering technique in OCaml but this represents a huge amount of work.

On the contrary, the re-implementation of the DBSCAN algorithm in OCaml (while using the MATLAB implementation that we worked on before as blueprint), and the programming of some simple techniques to process its results represents a smaller amount of work.

We therefore decide to build our own system that would use the good performance of the DBSCAN algorithm in low-dimensional spaces in a splitting step as exposed in section 3.2.4 and, on top of this, combine these clustering results with simple techniques based on the notion of *clustering ensemble* [137] as addressed in section 3.2.5.

We however think that the algorithms presented in section 2.1 could be of great interest in order to improve scalability issues (cf. section 6.2.2).

3.2.3 General presentation

UNADA is based on clustering techniques applied to the feature space $\mathbf{X}$. $\mathbf{X}$ contains F rows, one for each of the aggregated flows in the flagged time slot. Each row is composed of A values that represent the A traffic features. The objective of clustering is to partition a set of unlabeled samples into homogeneous groups of similar characteristics or *clusters*, based on a measure of similarity. Samples that do not belong to any of these clusters are classified as *outliers*. Our particular goal is to identify both those clusters and outliers. The most appropriate approach to find outliers is, ironically, to properly identify clusters.

We have developed a divide and conquer clustering approach, using the notions of clustering ensemble [137] and multiple clustering combinations. The idea is novel and appealing: why not taking advantage of the information provided by multiple partitions of $\mathbf{X}$ to improve clustering robustness and identification of clusters and outliers ? A clustering ensemble $\mathbf{P}$ consists of a set of multiple partitions P_i produced for the same data. Each of these partitions provides a different and independent evidence of data structure, which can be combined to construct a global partition that reflects natural groupings, clusters, and outliers. There are different ways to produce a clustering ensemble. We use Sub-Space Clustering (SSC) to produce multiple data partitions, doing density-based clustering in N different sub-spaces $\mathbf{X}_n$ of the original space $\mathbf{X}$. We refer the reader to the left part of Figure 3.3 to better understand this approach. Once these N partitions have been obtained, we apply a method of our own design in order to combine them and thus build a single general partition over the whole feature space. This method is located on the right of Figure 3.3.

3.2.4 Splitting with Sub-Space Clustering

In order to use unbiased attribute values for clustering, we normalize $\mathbf{X}$ into $\mathbf{X}_{norm}$. Each of the N sub-spaces $\mathbf{X}_n \subset \mathbf{X}_{norm}$ is obtained by selecting k features from the complete set of A attributes. To deeply explore the complete feature space, the number of sub-spaces N that are analyzed corresponds to the number of k-combinations-obtained-from-A. Each partition P_n is obtained by applying DBSCAN [29] to sub-space $\mathbf{X}_n$. The

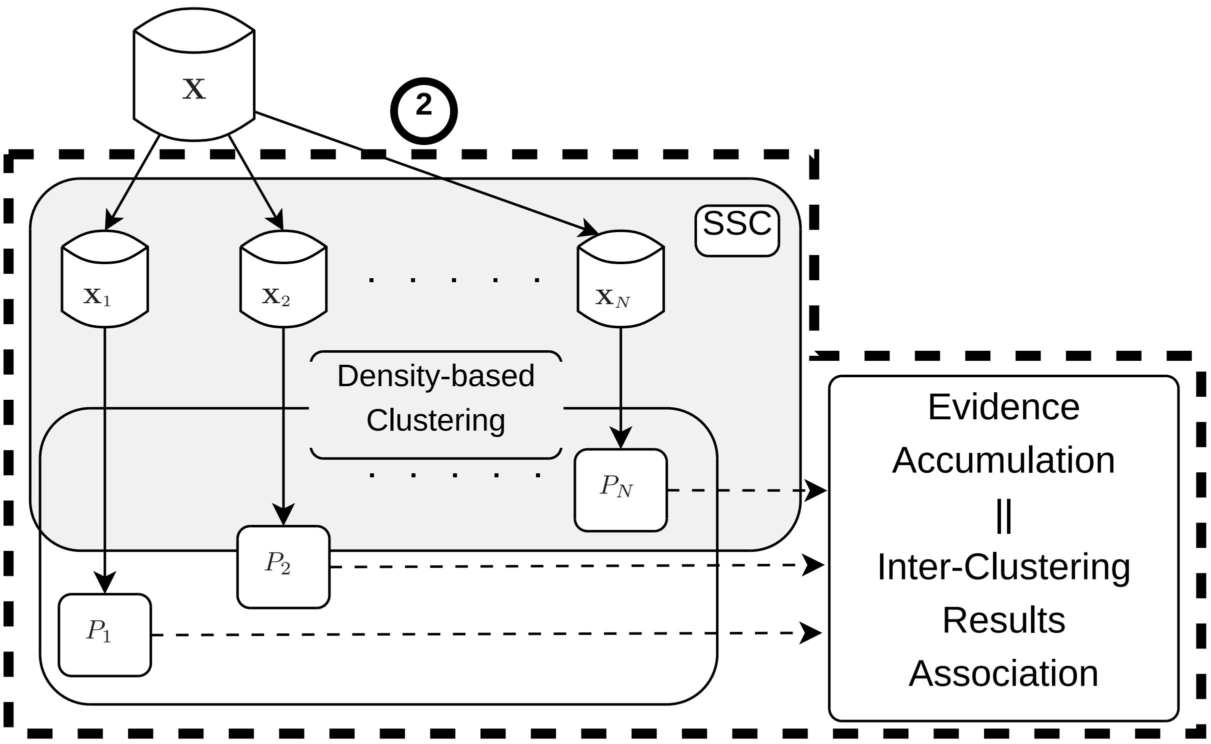


Figure 3.3: High-level scheme of the unsupervised learning technique used

choice of the clustering algorithm is motivated by the fact that the number of anomalies is partially linked to the number of found clusters (cf. anomaly representations in section 3.2.1). Clustering algorithms that need an alleged number of clusters, such as k-means, are therefore only able to find a predetermined number of anomalies. However, such number is impossible to determine empirically because of the inner unpredictable nature of network anomaly. DBSCAN, by not needing a-priori difficult to set parameters such as the number of clusters to identify, thus perfectly fits our needs. DBSCAN is a powerful clustering algorithm that discovers clusters of arbitrary shapes and sizes [138], relying on a density-based notion of clusters: clusters are high-density regions of the space, separated by low-density areas. Results provided by applying DBSCAN to sub-space $\mathbf{X}_n$ are twofold: a set of p^n clusters $\{C_1^n, C_2^n, .., C_{p^n}^n\}$ and a set of q^n outliers $\{o_1^n, o_2^n, .., o_{q^n}^n\}$. To set the number of dimensions k of each sub-space, we take a very useful property of monotonicity in clustering sets, known as the downward closure property (cf. property 3.1).

Property 3.1 *If a collection of elements is a cluster in a k-dimensional space, then it is also part of a cluster in any $(k-1)$ projections of this space. [38]*

This property directly implies that, if there exists any interesting evidence of density in $\mathbf{X}_{norm}$, it will certainly be present in its lowest-dimensional sub-spaces. Using small values for k provides several advantages: firstly, doing clustering in low-dimensional spaces is more efficient and faster than clustering in bigger dimensions. Secondly, density-based clustering algorithms such as DBSCAN provide better results in low-dimensional spaces [38], because high-dimensional spaces are usually sparse, making it difficult to distinguish between high and low density regions. Finally, clustering multiple low-dimensional subspaces provides a finer-grained analysis, which improves the ability of UNADA to detect anomalies of very different characteristics. We shall therefore use $k = 2$ for SSC. The number of subspaces N is equal to k-combinations-obtained-from-A or $\binom{k}{A}$. This thus gives:

$$\begin{aligned} N &= \binom{k}{A} \\ &= \frac{A!}{k!(A-k)!} \\ &= \frac{A!}{2!(A-2)!} \\ &= \frac{A(A-1)(A-2)...(A-(A-2))(A-(A-1))}{2(A-2)...(A-(A-2))(A-(A-1))} \\ &= \frac{A(A-1)}{2} \end{aligned}$$

In our case, the number N of subspace $\mathbf{X}_n$ is $N = \frac{A(A-1)}{2}$.

3.2.5 Combining step

Having produced the N partitions, the question now is how to use the information provided by the partitions. We here present the two strategies that we devised in order to achieve this goal. This step is depicted on the right of Figure 3.3.

3.2.5.1 Evidence Accumulation

A possible answer is provided in [139], where authors introduced the idea of Evidence Accumulation Clustering (EAC). EAC uses the clustering results of multiple partitions P_n to produce a new inter-patterns similarity measure which better reflects natural groupings. The underlying assumption in EAC is that patterns belonging to a "natural" cluster are likely to be co-located in the same cluster in different partitions. The rationale of EAC is the same for outliers: patterns that are outliers in one subspace are likely to be outliers in other subspaces. The principles of EAC have been published in [129, 130].

We shall adapt the EAC algorithm for our particular problem of unsupervised anomaly detection. For doing so, let us think about the particular structure of any general anomaly. According to section 3.2.1, an anomaly may consist of either outliers or small-size clusters, depending on the aggregation level of flows in $\mathbf{Y}$.

We implement two different EAC methods to isolate small-size clusters and outliers: EA for small-Clusters identification (EA4C), and EA for Outliers identification (EA4O). Algorithm 1 presents the pseudo-code for both methods. In EA4C, we assign a stronger similarity weight when patterns are assigned to small-size clusters. Taking the membership of pairs of patterns to the same cluster as weights for their association, N partitions are mapped into a $F \times F$ similarity matrix S, such that $S(f, g)$ contains the similarity between patterns (here flows) f and g. The weighting function $w_d(nb_n(d))$ used to update

Algorithm 1 EA4C & EA4O for Unsupervised Anomaly Detection

```
1: **Initialization:**
2:     Set similarity matrix $S$ to a null $F \times F$ matrix.
3:     Set dissimilarity vector $D$ to a null $F \times 1$ vector.
4: **for** $n = 1 : N$ **do**
5:     Update $S(f,g)$, $\forall$ pair $\{\mathbf{x}_f, \mathbf{x}_g\} \in C_d$ and $\forall\, C_d \in P_n$:
6:         $w_d \leftarrow e^{-\gamma \frac{(nb_n(d) - nbpDBS)}{F}}$
7:         $S(f,g) \leftarrow S(f,g) + \frac{w_k}{N}$
8:     Update $D(f)$, $\forall$ outlier $o_f \in P_n$:
9:         $w_n \leftarrow \frac{F}{(F - nb_n^{max}) + \epsilon}$
10:        $D(f) \leftarrow D(f) + \mathrm{d_M}(o_f, C_n^{max})\, w_n$
11: **end for**
```

$S(f,g)$ at each iteration n takes bigger values for small values of $nb_n(d)$. $nb_n(d)$ is the number of flows inside the d^{th} cluster which contains the pair $\{\mathbf{x}_f, \mathbf{x}_g\}$ located in subspace $\mathbf{X}_n$. We thus have $1 \leq d \leq p^n$, p^n being the number of cluster in partition P_n. $w_d(nb_n(d))$ goes to zero for big values of $nb_n(d)$. Note that if a pair of patterns $\{\mathbf{x}_f, \mathbf{x}_g\}$ is assigned to the same cluster in each of the N partitions then $S(f,g) = 1$, which corresponds to maximum similarity. The parameter *nbpDBS* specifies the minimum number of flows that can be classified as a cluster by the DBSCAN algorithm. The parameter γ permits to set the slope of $w_k(nb_n(k))$.

In the case of EA4O, we define a dissimilarity vector D where the distances from all the different outliers to the centroid of the biggest cluster identified in each partition P_n are accumulated. We shall use C_n^{max} as a reference to this cluster. The idea is to clearly highlight outliers that are far from the normal-operation traffic in the different partitions, statistically represented by C_n^{max} as stated in section 3.2.1. The weighting factor w_n takes bigger values when the size nb_n^{max} of C_n^{max} is closer to the total number of patterns F, meaning that outliers are more rare and become consequently more important . The parameter ϵ is simply introduced to avoid numerical errors ($\epsilon = 1e^{-3}$). Finally, instead of using a simple Euclidean distance, we compute the Mahalanobis distance $\mathrm{d_M}(o_f, C_n^{max})$ between the outlier and the centroid of C_n^{max}.

The Euclidean distance between two points $p = (p_1, p_2, ..., p_n)^T$ and $q = (q_1, q_2, ..., q_n)^T$ is :

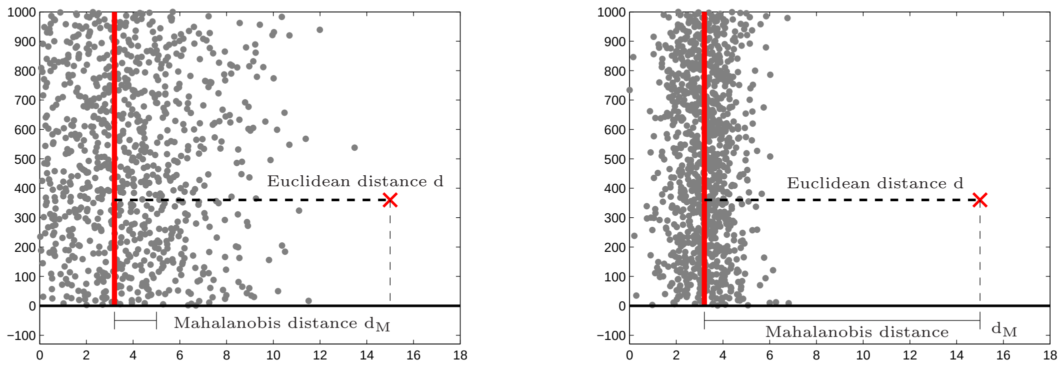
$$D_E(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{3.1}$$

The Mahalanobis distance of a vector $x = (x_1, x_2, x_3, \ldots, x_N)^T$ is

$$D_M(x) = \sqrt{(x-\mu)^T S^{-1} (x-\mu)} \tag{3.2}$$

$$\text{where : } \begin{cases} \mu = (\mu_1, \mu_2, \mu_3, \dots, \mu_N)^T : & \text{the vector containing the mean of the considered values along each } N \text{ dimensions} \\ S : & \text{the covariance matrix.} \end{cases} \tag{3.3}$$

Figure 3.4 highlights the difference between Euclidean and Mahalanobis distances. The Mahalanobis distance takes into account the variance of instances along a dimension. Its value is small when points are spread along the considered dimension (cf. Figure 3.4a). Corollary, the Mahalanobis distance is bigger when the majority of the points are more concentrated in the feature space (cf. Figure 3.4b). While in both cases, i.e. in Figure 3.4a and 3.4b, the Euclidean distance stays the same.



(a) (b)

Figure 3.4: Euclidean vs. Mahalanobis distance as a measure of dissimilarity.

In the final step, any clustering algorithm can be applied to matrix S or to vector D to obtain a final partition of $\mathbf{X}_{norm}$ that isolates both small-size clusters and outliers. As we are only interested in finding the smallest-size clusters and the most dissimilar outliers, the detection consists in finding the flows with the biggest similarity in S and the biggest dissimilarity in D. This is simply achieved by comparing the values in S and D to a variable detection threshold. This threshold is to be further tuned by the operator in order to obtain a satisfying trade-off between True Positive Rate (TPR) and False Positive Rate (FPR).

3.2.5.2 Inter-Clustering Results Association

However, by reasoning over the similarities between patterns (here flows), EAC introduces several potential errors. A first possible error is linked to the use of a wrong parameter in a sensitive clustering algorithm over S values. Furthermore, the use of a threshold over S and/or D can decrease the system performance in case of a wrong value used. Another potential error in the EA4C algorithm, is due to its algorithm. Let us consider two flow sets or clusters C_d and C_e. If the cardinality of these flow sets is close and if they are present in a similar number of sub-spaces, then, EA4C will produce a very close similarity value for all patterns in both flow sets. They will then likely be falsely considered as belonging to the same cluster. This possibility is to be considered very seriously as it can induce a huge error: different anomalies will be merged together in a single global cluster in $\mathbf{P}$ and will then likely be wrongly identified and, even more detrimental to our system, wrongly characterized.

In order to avoid the previously exposed sources of error, we devise another way of combining clustering results obtained from sub-spaces: Inter-Clustering Results Association. The rationale of this new method is to address the partition combination problem in terms of cluster of flows and outlier similarities instead of pattern (or flow) similarity. Hence, we shift the similarity measure from the patterns to the clustering results (clusters and outliers). The underlying idea is straightforward: identify clusters or outliers present in different sub-spaces that contain the same flows. The problem can then be split in two sub-problems: correlate clusters through Inter-CLuster Association (ICLA), and correlate outliers through Inter-Outlier Association (IOA). These concepts have been published in [131].

In each case, cluster similarity (ICLA) or outlier similarity (IOA), a graph is used to express similarity between either clusters or outliers. Each vertex represents a cluster or an outlier in any sub-space $\mathbf{X}_n$ and each edge represents the fact that two connected vertices, and hence, clusters or outliers, are similar.

Let CS be the cluster similarity measure between two clusters C_d and C_e:

$$CS(C_d, C_e) = \frac{|C_d \cap C_e|}{max(|C_d|, |C_e|} \tag{3.4}$$

where $|C|$ is the cardinality of the cluster C, and $C_d \cap C_e$ the intersection of C_d and C_e.

Each edge in the *cluster similarity graph* used for ICLA between two clusters C_d and C_e means $CS(C_d, C_e) > tICSim$. $tICSim$ is the threshold on Inter-Cluster Similarity. $tICSim$ optimal value is further empirically evaluated on real network traffic (cf. section 5.2.3.3). IOA uses an *outlier similarity graph* built by linking every outlier to every other outlier that contains the same pattern.

We now need to devise a technique to analyze the similarity graphs. Let's first look at ICLA. The problem definition states that we need to find clusters found in different subspaces $\mathbf{X}_n$ that contain same flows sets. Considering the cluster similarity graph, we need to find vertex sets where every vertex is linked to every other vertices. In graph theory, a *clique* in an undirected graph $G = (V, E)$, is defined as a subset C of the vertex set V where $C \subseteq V$, such that for every two vertices in C, an edge connects the two considered vertices. The searched vertex sets in our use case are thus the cliques inside cluster similarity graphs. Concerning IOA, the rationale is the same. The only difference is that similarity is processed over a single pattern/flow. This means that we do not use a similarity threshold for the outlier similarity graph. The previously exposed relation between clusters sets and cliques also applies here to outliers sets and cliques in the outlier similarity graph.

These cliques could be easily extracted from cluster and outlier similarity graphs by a simple heuristic that would use a maximum clique search algorithm. We could launch several successive execution maximum clique search algorithm and remove every edges of each found maximum clique from the considered similarity graph. This would allow us to find every maximum cliques inside each similarity graphs. However, the maximum clique search problem NP-complete [140]. Existing solution are thus too slow for our application. We then make the hypothesis that a vertex can only be part of a single clique. The rationale behind this hypothesis is that it allows us to use a greedy algorithm to find maximum non-intersecting cliques inside similarity graphs. Such greedy algorithm are much faster than classic algorithm.

We verify the pertinence of use of such greedy algorithms by verifying the high probability of verification of the previously proposed hypothesis over real data graphs obtained from several runs of our algorithm over real packet traces. First, we show that a majority of connected component inside similarity graphs are also cliques. In graph theory, connected component of an undirected graph are subgraphs where every vertex pair is connected through a path. This guarantees that every vertex located in a connected component belongs to exactly one clique which is the considered connected component. We analyze cluster and outlier similarity graphs generated by several execution of UNADA. Our algorithm is thus applied upon the traffic traces used for sensitivity evaluation (i.e. one trace by month from January 2001 to December 2006 of the MAWI repository [4], cf. section 5.2.3). We here use two aggregation levels: source IP address with /24 (l_3) and destination IP address with /24 (l_6). We thus obtain two graphs for each aggregation levels used for every time slot flagged as anomalous by the change-detection step. UNADA therefore build four graphs: cluster similarity graph for l_3, cluster similarity graph for l_6, outlier similarity graph for l_3 and outlier similarity graph for l_6. Figure 3.5a displays the proportion of connected components that are also cliques for each of the four generated graphs. It clearly shows that an overwhelming percentage of connected components are also cliques. One can note that connected components in the outlier similarity graphs (the two bar chart on the left) are always cliques. This phenomenon is due to the binary nature of outlier similarity. Outliers contain exactly one flow. A vertex representing an outlier is thus either similar or different from any vertex of a connected component in outlier similarity graphs. This first analysis guarantees us that a vast majority of connected components are also cliques. Every point in these connected components therefore only belongs to a single clique. This verifies our initial hypothesis.

Second, we study the case of connected components which are not cliques. The classical solution would be to use exhaustive search of cliques inside the connected component to find maximum cliques. As said in previous paragraphs, this solution is not possible because of its complexity. We therefore run a greedy algorithm over each connected component that is not a clique and analyze the ratio of point contained in the found clique over the number of point in the considered connected component. Figure 3.5b is built upon data extracted from the dataset used in section 5.2. It shows that the mean ratio of the number of vertices in the maximum clique found inside each connected component over the number of vertices of the considered connected component is slightly over 0.85. It allows us to assess that, even if a connected component is not a clique, the vast majority of its vertices are captured by the maximum clique found by our greedy algorithm.

The first of these two analyses verifies that the hypothesis quoted above (a vertex inside a similarity graph belongs to a single clique) exhibits has a very high probability of realization. The second analysis addresses the scenario where this hypothesis is not verified. This analysis proves that even if a connected component is not a clique per se, our algorithm manage to capture a vast majority of its vertices in a clique.

Finally, similar flow sets are extracted from cliques found inside similarity graphs. Such flows sets are also called traffic segments. For cluster similarity graphs, flow sets

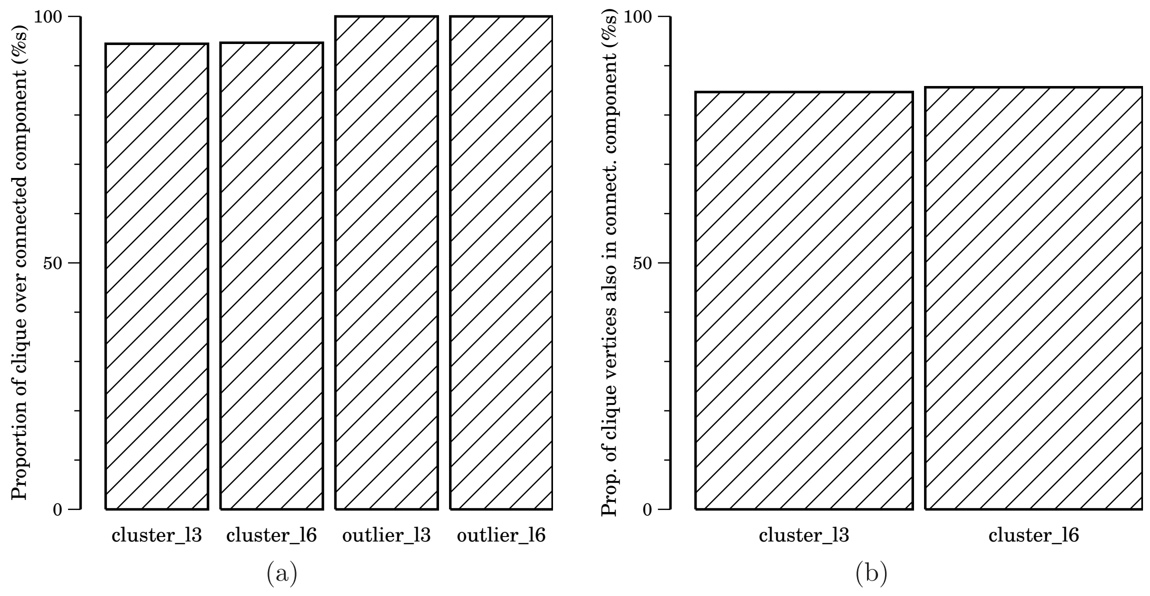


Figure 3.5: Proportion of connected components who are also cliques over connected component inside similarity graphs (a) and proportion of vertices in biggest clique inside a connected component over number of vertices in the same connected component when the considered connected component is not a clique inside similarity graphs (b).

contained in each traffic segments are the intersection of all the flow sets present in each clusters of a clique. The same rationale is applied to outlier similarity graphs. The only difference is that flow sets of traffic segment extracted from outliers contain only one flow. We therefore do not use intersection of flow sets for cliques extracted from outlier similarity graphs. Flows that do not belong to, neither traffic segments of clusters, nor traffic segments of outliers, are considered as noise. They are thus not further processed by our algorithm since we cannot asses their normality or abnormality.

3.2.6 Normal traffic representation redefinition

In the light of the clustering technique internals exposed in sections 3.2.3, 3.2.4 and 3.2.5, and in order to ease the automation of the anomaly detection process, we redefine the representation of the normal traffic in the feature space defined in hypothesis 3.1 as follows:

Hypothesis 3.2 *The majority of aggregated flows are normal.*
These normal flows form a single cluster *accounting for at least* $Y\%$ *of aggregated flows.*

The downward closure property also allows us to extend the definition of the normal traffic representation addressed in section 3.2.1. Since we now assume that normal traffic is represented by one cluster containing at least half of the F flows, the property guarantees us that this cluster will also be present in each $\mathbf{X}_n$ subspaces. We therefore enforce that this property is verified at two points in our algorithm and thus derive two hypothesis from the previous one.

Hypothesis 3.3 *In each subspace* $\mathbf{X}_n$*, the biggest cluster (in terms of number of aggregated flows)* must *contain at least Y% of aggregated flows.*

Hypothesis 3.4 *In each feature space, the biggest traffic segment (in terms of number of aggregated flows)* must *account for at least Y% of the total aggregated flows number.*

If either one of these hypotheses is not verified, we abort the analysis and resume the execution of UNADA on the next time slot. The consequences of this hypothesis are discussed in section 6.2.1.

3.2.7 Summary

Our algorithm uses a novel clustering technique to extract anomalous flows from the whole traffic. Anomalies and normal traffic have specific representations inside the feature space. According to the hypothesis 1.1 and 3.2, normal traffic is always a single cluster accounting for at least half of the aggregated flows, anomalies, on the other hand, can be represented as outliers or clusters according their distributed nature and the used aggregation level. The clustering technique used relies on the notion of clustering ensemble. It applies the DBSCAN algorithm into several subspaces of the original feature space to produce several partitions. These partitions are then combined to build a global partition.

The analysis of the computational time presented in the next section is then critical in order to asses the scalability of UNADA for real-world use.

3.3 Computational time analysis

This section provides insights in the behavior of our algorithm regarding the computational time. We analyze the Computational Time (CT) of UNADA. The SSC-EA/Inter-Clustering Results Association (ICRA)-based algorithm performs multiple clusterings in $N(A)$ low-dimensional sub-spaces $\mathbf{X}_n \subset \mathbf{X}_{norm}$, A being the number of attributes used. These multiple computations impose scalability issues for on-line detection of attacks in very-high-speed networks. Two key features of the algorithm are exploited to reduce scalability problems in number of features A and the number of aggregated flows F to analyze. Firstly, clustering is performed in very-low-dimensional sub-spaces, $\mathbf{X}_n \in \mathbb{R}^2$, which is faster than clustering in high-dimensional spaces [138]. Secondly, each sub-space can be clustered independently of the other sub-spaces, which is perfectly adapted for parallel computing architectures. Parallelization can be achieved in different ways: using a single multi-processor and/or multi-core machine, using network-processor cards [7] and/or Graphic Processor Unit (GPU) capabilities, using a distributed group of machines, or combining these techniques. We shall use the term "slice" as a reference to a single computational entity. We first expose an insight on UNADA execution time and speedup through parallelism, and then, we provide a computational analysis on the clustering time of our algorithm.

[7]http://www.tilera.com/products/platforms

3.3.1 UNADA execution time analysis

The UNADA implementation is made in OCaml. We use the Functory library [141] to implement parallel distributed clustering of the N subspaces. This library provides a generic interface and allows the final user to easily switch between sequential execution, locally parallelized execution between several cores or processors on a single machine (called "Cores" in Functory) and parallelized execution on several machines (called "Network" in Functory). The "Network" execution mode uses two executables: one managing the processing called "Master" and one processing the data called "Worker". The "Network" execution mode includes three different scenarios: same executables between "Master" and "Worker", different executables but same version of OCaml and different executables and different versions of OCaml. The closer the "Master" and the "Worker" executables are (depending on the OCaml version and the source code), the easier for OCaml to serialize data to be transfered between master and worker. These three scenarios offer different features to the user according to his experimental setup.

In our case, we use the network setup with different executables between master and worker. A first machine uses an executable that runs the UNADA algorithm and manages the clustering step. Another machine clusters the data through a very simple executable when the first one needs to do so. We choose to use two different executables for several reasons. First, UNADA uses many packages and some are manually built libraries such as Admd (cf. section 5.2.1). The machine running the "Worker" executable being used by someone on a daily basis, we want to avoid useless experimental libraries installation. Second, the use of a different executable between the master and the worker, allow us to modify the master executable without updating the "Worker" code. This is especially important for debugging tasks. The "Worker" machine that proceed to the clustering is a desktop machine constituted of an Intel i7 860 of 4 cores with Hyperthreading, and of 8GB of RAM.

Figure 3.6 shows the execution time proportions between source and destination aggregated flows clustering time and other processing time according to the number of used slice or processes. The execution time is defined as the processing of the time slot from the 1st to the 15 th second of the trace of January 1st 2009 of the MAWI repository [4]. The clustering time of source aggregated flows is greater than the clustering time of destination aggregated flows for any number of used process. This is caused by the fact that, for this particular time window, the number of source aggregated flows is greater than the number of destination aggregated flows: 8995 to 6160. This figure clearly shows that clustering is the most time-consuming task of the UNADA algorithm. It is therefore pertinent to focus our computational time analysis on clustering time.

Figure 3.7 details the original and normalized clustering time for several executions using one to eight slices. The theoretical curve represents the optimal speed-up: the nominal clustering time (sequential clustering of the N subspaces) divided by the number of process used. The speed-up is very close to the theoretical value for the four first points. This proves that the parallel clustering on several slices is useful. However, several observations can be made. First, there is a little difference for the four first points. This can be explained by the fact that the processor may become less efficient when more and more processes are used. Second, the speedup after the fourth point (i.e. when we use more than four processes) is far from what is expected. the most probable

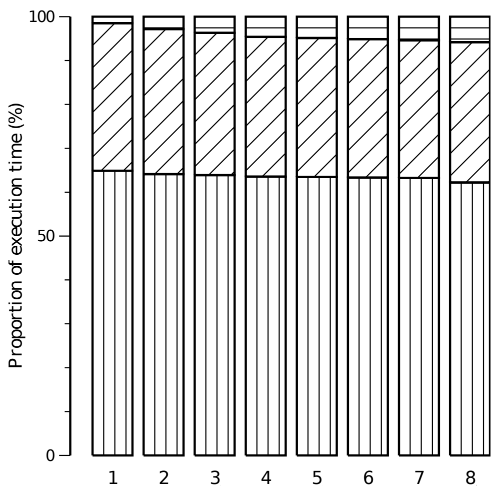


Figure 3.6: Proportion of execution time for source aggregated flows clustering (vertical stripes), destination aggregated flows clustering (diagonal stripes) and other processing (horizontal stripes) regarding the number of slice (processes) used.

reason for this behavior is the inability of Hyperthreading to actually provide two fully working cores out of a single one. The slight measured speedup may be caused by the reduced idle time between atomic tasks caused by Functory's communications. In fact, if UNADA uses more than four processes, once a job is finished on one of the four processes, any other job can immediately be processed on the free process, while it is not the case if the program only processes four or less tasks at a time.

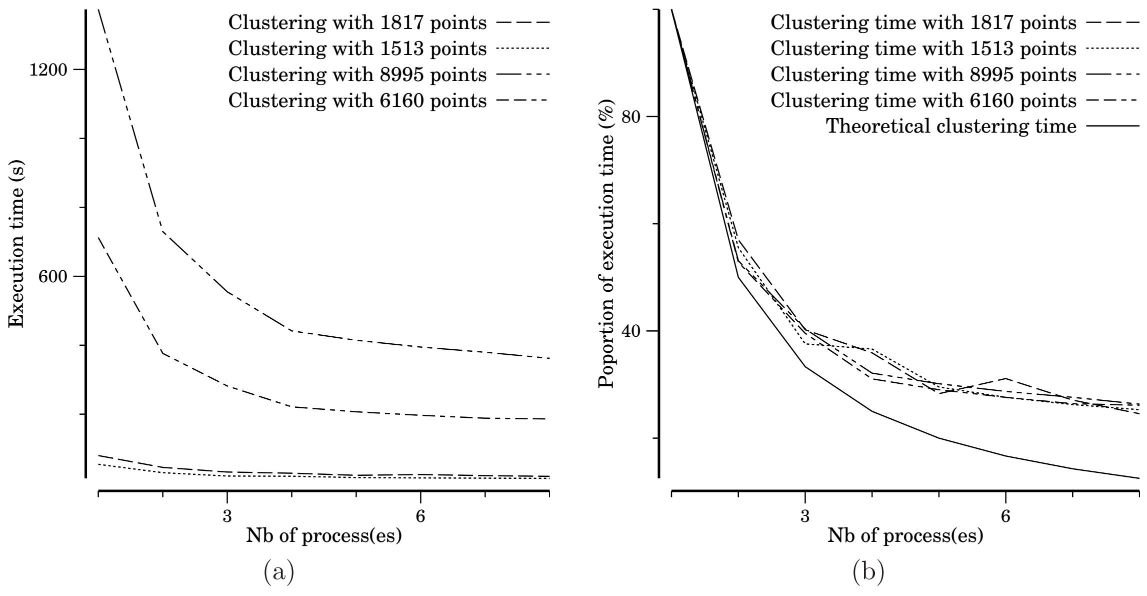


Figure 3.7: Clustering time and normalized clustering time as a function of number of processes used.

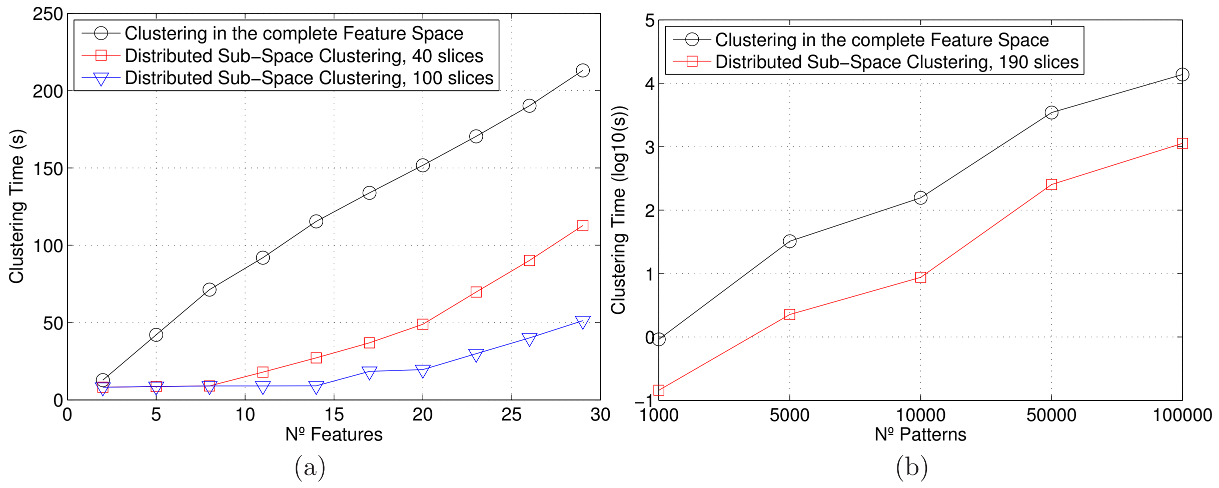


Figure 3.8: Computational Time as a function of the number of features and number of flows to analyze. The number of aggregated flows in (a) is $F = 10000$. The number of features and slices in (b) is $A = 20$ and $S = 190$ respectively.

3.3.2 Clustering time simulation

We here analyze the clustering time of UNADA. This analysis has been published in [127]. We use a large number of aggregated flows, $F = 10^4$, and use two different numbers of slices, $M = 40$ and $M = 100$. The analysis is done with traffic from the WIDE network, combining different traces to attain the desired number of flows. To estimate the CT of SSC for a given value of A and M, we proceed as follows: first, we separately cluster each of the $N = A(A-1)/2$ sub-spaces $\mathbf{X}_n$, and take the worst-case of the obtained clustering time as a representative measure of the CT in a single sub-space, i.e., $\text{CT}(\mathbf{X}_{norm}^{\text{SSCwc}}) = \max_i \text{CT}(\mathbf{X}_n)$. Then, if $N \leqslant M$, we have enough slices to completely parallelize the SSC algorithm, and the total CT corresponds to the worst-case, $\text{CT}(\mathbf{X}_{norm}^{\text{SSCwc}})$. On the contrary, if $N > M$, some slices have to cluster various sub-spaces, one after the other, and the total CT becomes $(N//M + 1)$ times the worst-case $\text{CT}(\mathbf{X}_{norm}^{\text{SSCwc}})$, where $//$ represents integer division.

Figure 3.8 depicts the CT of the SSC-based algorithm, both (a) as a function of the number of features A used to describe traffic flows and (b) as a function of the number of flows A to analyze. Figure 3.8a compares the CT obtained when clustering the complete feature space $\mathbf{X}_{norm}$, referred to as $\text{CT}(\mathbf{X}_{norm})$, against the CT obtained with SSC, varying A from 2 to 29 features. The first interesting observation from Figure 3.8b regards the increase of $\text{CT}(\mathbf{X}_{norm})$ when A increases, going from about 8 seconds for $A = 2$ to more than 200 seconds for $A = 29$. As we said before, clustering in low-dimensional spaces is faster, which reduces the overhead of multiple clusterings computation. The second paramount observation is about parallelization: if the algorithm is implemented in a parallel computing architecture, it can be used to analyze large volumes of traffic using many traffic descriptors in an on-line basis; for example, if we use 20 traffic features and a parallel architecture with 100 slices, we can analyze 10000 aggregated flows in less than 20 seconds.

Figure 3.8b compares $\text{CT}(\mathbf{X}_{norm})$ against $\text{CT}(\mathbf{X}_{norm}^{\text{SSCwc}})$ for an increasing number of flows F to analyze, using $A = 20$ traffic features and $M = N = 190$ slices (i.e., a completely parallelized implementation of the SSC-based algorithm). As before, we can appreciate the difference in CT when clustering the complete feature space vs. using low-dimensional sub-spaces: the difference is more than one order of magnitude, independently of the number of flows to analyze. Regarding the volume of traffic that can be analyzed with this 100% parallel configuration, the SSC-based algorithm can analyze up to 50000 flows with a reasonable CT, about 4 minutes in this experience. In the presented evaluations, the number of aggregated flows in a time slot of $\Delta T = 20$ seconds rounds the 2500 flows, which represents a value of $\text{CT}(\mathbf{X}_{norm}^{\text{SSCwc}}) \approx 0.4$ seconds. For the $A = 9$ features that we have used ($N = 36$), and even without doing parallelization, the total CT is $N \times \text{CT}(\mathbf{X}_{norm}^{\text{SSCwc}}) \approx 14.4$ seconds.

3.3.3 Summary

We first use the OCaml implementation of UNADA to experimentally prove that clustering is by far the most time consuming task of our algorithm and thus, the prime concern in order to study the scalability of our algorithm. We then demonstrate that parallelism is possible and greatly improves the computational time of UNADA. We also show that our algorithm displays a good scalability regarding number of flows and number of attributes.

3.4 Summary

In this chapter, we have presented our unsupervised anomaly detection algorithm. UNADA uses two hypotheses: the anomalous traffic is statistically different from the normal one and the normal traffic is bigger than the anomalous one. Our work relies on these hypotheses to design an unsupervised learning algorithm able to isolate anomalous flows from normal ones. We state that both normal traffic and anomalous traffic display very specific and different characteristics in the feature space and can therefore be separated. We developed a novel approach based on clustering that produces a global partition for the whole feature space. Our algorithm uses two successive steps to build this global partition: it first splits the feature space into many subspaces in order to build reliable partitions. It then combines these partitions into a single global one.

Finally, the computational time of our algorithm is theoretically and empirically measured. UNADA scales reasonably well regarding the number of flows and the number of attributes. Our algorithm can thus be distributed with a very good speedup.

Once anomalies are extracted, a post-processing step is needed in order to reduce the workload of the operator. This step is addressed in the next chapter and aims at reducing the number of anomalies, ranking anomalies by dangerousness and characterizing anomalies.

Chapter 4

Anomaly post-processing

The unsupervised clustering step exposed in chapter 3, allows us to extract anomalies from a feature space $\mathbf{X}$ obtained from F aggregated flows according to an aggregation level l_i. However the potential number of detected anomalies can be high. The danger is here to overwhelm the operator with a storm of alarms. A post-processing step is thus crucial to help the operator to prioritize its own work towards the most dangerous anomalies. A more general help shall also be provided to the operator concerning tedious tasks such as investigation on anomaly nature. Unfortunately, at this point, our algorithm does not provide any method to cope with these problems.

We thus propose a three steps post-processing technique that reduces the amount of work provided by the operator. First, we reduce the overall number of found anomalies by correlating anomalies extracted from aggregated flows obtained for different aggregation levels l_i. Second, we provide a technique that allows the operator to prioritize its work through ranking anomalies by dangerousness. Third, we present heuristics to build anomaly signatures that spare to the operator the anomaly analysis. Each of these steps are depicted by Figure 4.1.

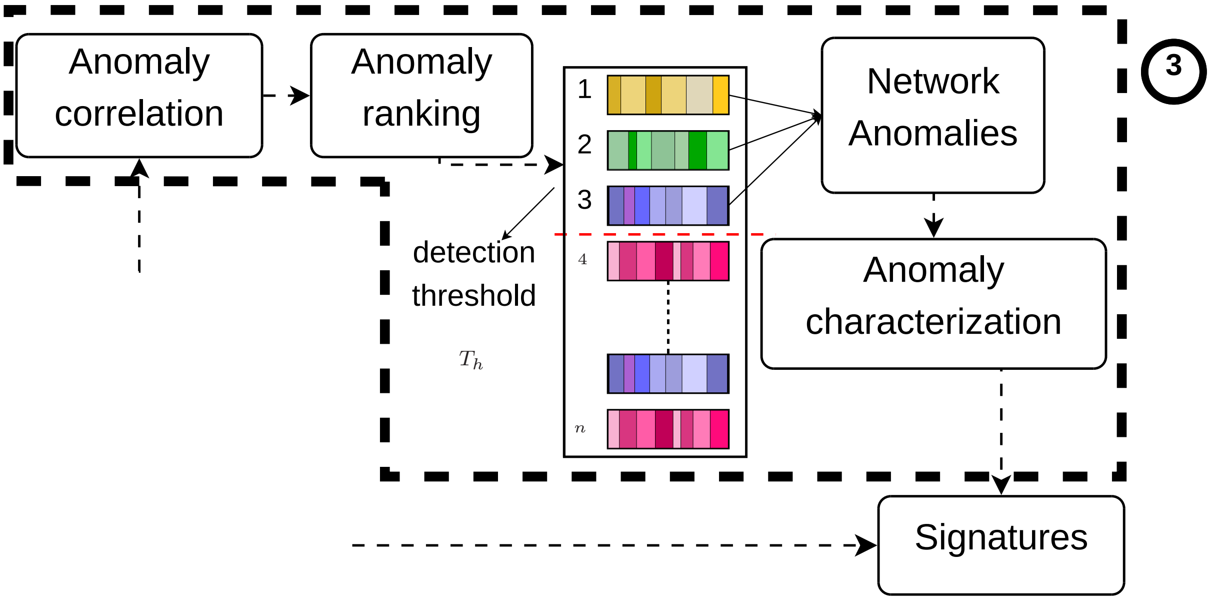


Figure 4.1: High-level scheme of the anomaly post-processing step.

4.1 Anomaly correlation

In the previous chapter, we showed that our unsupervised algorithm is capable of extracting anomalies from flows aggregated according to a specific aggregation level l_i. Our algorithm may be used on several different aggregation levels and will thus generate a huge number of alarms. The interrogation that arises here is: why not trying to correlate anomalies extracted from flows obtained from different aggregation levels l_i in order to reduce the number of alarms while improving overall reliability ? We therefore follow this rationale by obtaining several clustering results, each of them built from several feature spaces related to different aggregation levels, and devising a strategy to correlate anomalies extracted from these clustering results. This anomaly correlation method has been published in [131].

In order to correlate anomalies found in different aggregation levels, we define two unique anomaly characteristics: its source and its destination. We refine this definition by associating an anomaly source with its source IP address set and an anomaly destination with its destination IP address set.

Let $Sim_{IP@}(S^1_{IP@1}, S^2_{IP@2})$ be the similarity between two IP address sets $S^1_{IP@}$ and $S^2_{IP@}$:

$$Sim_{IP@}(S^1_{IP@}, S^2_{IP@}) = \frac{|S^1_{IP@} \cap S^2_{IP@}|}{max(|S^1_{IP@}|, |S^2_{IP@}|)} \tag{4.1}$$

where $|S|$ is the cardinality of the set S.

Two anomalies T_1 and T_2 and their associated source IP address set, $S^1_{srcIP@}$ and $S^2_{srcIP@}$, and destination IP address set, $S^1_{dstIP@}$ and $S^2_{dstIP@}$, are similar if the two following equations are true:

$$Sim_{srcIP@}(S^1_{srcIP@}, S^2_{srcIP@}) > tTSAddrSim \tag{4.2}$$

$$Sim_{dstIP@}(S^1_{dstIP@}, S^2_{dstIP@}) > tTSAddrSim \tag{4.3}$$

where $tTSAddrSim$ is the threshold on address set similarity.

In other words, two anomalies are similar if both their sources and destinations are similar regarding a threshold called tTSAddrSim.

In this work, we only correlate anomalies detected from two types of aggregation: IP source aggregation and IP destination aggregation. We hence avoid correlating anomalies located in the same aggregation level types (e.g. l_1 and l_2), that would be potentially contained in each other. Moreover, we only use the /24 aggregation level. Both of these choices are made in order to reduce the amount of processing. The used aggregation levels are thus l_3 and l_6 (cf. aggregation levels definition in section 3.1.2). Section 6.2.3.1 provides prospectives concerning the use of more aggregation levels.

The final set of correlated anomalies is obtained from each couple of similar anomalies.

4.2 Anomaly ranking

In order to increase its own efficiency, a network operator analyzing occuring anomalies needs to prioritize its work towards the most dangerous anomalies. In the field of network anomaly detection, the more an anomaly is dangerous, the more its mitigation is critical. Thus, task prioritization shall be realized by ranking anomalies according to their dangerousness.

After the anomaly correlation step, many anomalies have been detected and some have also been correlated. Correlated anomalies provide us with a prima facie evidence of the anomalies hierarchy regarding their dangerousness. In fact, if an anomaly appears as such within several aggregation levels, it means that its flows are significantly different from the normal traffic in each of these aggregation levels, and thus, potentially dangerous in each of them. Such anomalies can therefore be considered as more dangerous than anomalies that are not correlated. This binary classification between correlated and uncorrelated anomalies, provides us a simple scheme of a possible anomaly dangerousness ranking.

In order to further improve this ranking, we introduce two other anomaly characteristics: its number of packets and its number of bytes. We thus build a dangerousness index that uses these two characteristics. We actually do not use the absolute number of packets and bytes but the proportion of traffic belonging to this anomaly in terms of number of packets and bytes.

Let DI be the Dangerousness Index built according to the three previously exposed criteria: detection in multiple feature space, proportion of packet number, proportion of byte number.

$$DI(Anom_1) = C * (P(\#pkts) + P(\#bytes)) \tag{4.4}$$

$$\text{where :} \begin{cases} \text{C : number of feature space(s) aggregation level(s) where the anomaly has} \\ \text{been detected} \\ P(\#pkts) = \frac{\#\text{pkts in anomaly}}{\#\text{pkts in time}-\text{slot}} \\ P(\#bytes) = \frac{\#\text{bytes in anomaly}}{\#\text{bytes in time}-\text{slot}} \end{cases} \tag{4.5}$$

This formula defines a basic dangerousness criterion which is the mean between the proportion of traffic belonging to the considered anomaly in terms of number of packets and the proportion of traffic belonging to the considered anomaly in terms of number of bytes. The Dangerousness Index of uncorrelated anomalies uses this basic formula. The Dangerousness Index of correlated anomalies is then obtained by simply multiplying this base value by the number of time the considered anomaly has been correlated. This step is designed to reflect the potential of dangerousness of an anomaly.

From a general point of view, the bigger $DI(Anom_1)$, the more dangerous the anomaly $Anom_1$. The T_h threshold in Figure 4.1 is a threshold on the DI value for the considered anomaly. This method has been partially published in [131].

4.3 Automatic anomaly characterization

At this stage, the algorithm has identified and ranked several anomalies constituted by a set of traffic flows in $\mathbf{Y}$ far out the majority of the traffic. The network operator can thus choose which anomalies should be processed first. However, the manual anomaly analysis task remains to be done. We here propose a technique to extract semantically interesting information about anomalies through filtering rules and signatures. This technique has been published in [130].

The first task is to automatically produce a set of K filtering rules $f_k(\mathbf{Y})$, $k = 1, .., K$ to correctly isolate and characterize anomalous flows. On one hand, such filtering rules provide useful insights on the nature of the anomaly, easing the analysis task of the network operator. On the other hand, rules can be combined to construct a signature of the anomaly, which can be used to detect its occurrence in the future, using a traditional signature-based detection system. Even more, this signature could eventually be compared against well-known signatures to automatically classify the anomaly [86, 142].

4.3.1 Rule building

In order to produce filtering rules $f_k(\mathbf{Y})$, the algorithm apprehends anomaly representation, cluster or outlier, inside the whole feature space. We define two different classes of filtering rule: *absolute* rules FR_A and *relative* rules FR_R.

4.3.1.1 Relative rules

Relative filtering rules are used in the characterization of both anomalous traffic segments. They depend on the relative separation between anomalous and normal-operation flows. Basically, if the anomalous flows are well separated from the normal clusters in a certain partition P_n built on a certain subspace $\mathbf{X}_n$, then the features of the corresponding subspace $\mathbf{X}_n$ are good candidates to define a relative filtering rule.

A relative rule defined for feature a and the $\mathbf{Y}_g$ flow set has the form ($||$ is the logical OR operator):

$$FR_R(\mathbf{Y}_g, a) = \{\forall \mathbf{y}_f \in \mathbf{Y}_g \subset \mathbf{Y} : x_f(a) < \lambda || x_f(a) > \lambda\} \tag{4.6}$$

In other words, a filtering rule FR_R is defined for a flow set $\mathbf{Y}_g$ on attribute a, $\mathbf{Y}_g$ being subset of the global flow set $\mathbf{Y}$. This filtering rule specifies that for each flow $\mathbf{y}_f$ belonging to $\mathbf{Y}_g$, the value of attribute a for this particular flow $\mathbf{y}_f$ will be greater or lower than a threshold λ.

We shall also define a *covering relation* between filtering rules. Let $FR^1_R(\mathbf{Y}_g, a)$ and $FR^2_R(\mathbf{Y}_g, a)$ two filtering rules ($||$ is the logical OR operator):

$$FR^1_R(\mathbf{Y}_g, a) = \{\forall \mathbf{y}_f \in \mathbf{Y}_g \subset \mathbf{Y} : x_f(a) < \lambda_1 || x_f(a) > \lambda_2\} \tag{4.7}$$

$$FR^2_R(\mathbf{Y}_g, a) = \{\forall \mathbf{y}_f \in \mathbf{Y}_g \subset \mathbf{Y} : x_f(a) < \lambda_3 || x_f(a) > \lambda_4\} \tag{4.8}$$

We say that:

$$\begin{aligned} &\text{Rule } FR^1_R(\mathbf{Y}_g, a) \text{ covers rule } FR^2_R(\mathbf{Y}_g, a).\\ &\Leftrightarrow FR^2_R(\mathbf{Y}_g, a) \subset FR^1_R(\mathbf{Y}_g, a)\\ &\Leftrightarrow \lambda_1 > \lambda_3 || \lambda_2 < \lambda_4 \end{aligned} \tag{4.9}$$

In others words, let's define two filtering rules FR^1_R and FR^2_R. FR^1_R specifies that the flow values for attribute a are lower than λ_1 (resp. greater than λ_2). FR^2_R specifies that the flow values for attribute a are lower than λ_3 (resp. greater than λ_4). FR_R^1 covers FR_R^2 if λ_1 is greater than λ_3 (resp. λ_2 is lower than λ_4).

If two or more rules from different partitions P_n overlap (i.e., they are associated to the same feature), the algorithm keeps the one that covers the rest.

The λ thresholds are built using data from the feature space. Let us consider a relative rule with a threshold λ_1 regarding a single attribute a. The λ_1 value is at half-distance between the representation of the normal traffic (a single cluster) and the representation of the abnormal traffic (i.e. either a cluster or an outlier). We actually use the mean of each representation. We thus take the mean of the attribute values for the considered attribute a for all the flows located in each representation. We then determine λ_1 by taking taking the point located at half-distance between the two means.

4.3.1.2 Absolute rules

Contrary to relative rules, absolute rules do not depend on the separation between normal and anomalous flows. They instead correspond to the presence of dominant features in anomalous flows. They are built upon every feature a for which no relative rule have been built.

An absolute rule for a certain feature a characterizing a certain flow set $\mathbf{Y}_g$ has the form (|| is the logical OR operator):

$$FR_A(\mathbf{Y}_g, a) = \{\forall \mathbf{y}_f \in \mathbf{Y}_g \subset \mathbf{Y} : x_f(a) = \lambda\} \tag{4.10}$$

One can also say that an absolute rule FR_A is defined for a flow set $\mathbf{Y}_g$ on attribute a, $\mathbf{Y}_g$ being subset of the global flow set $\mathbf{Y}$. This absolute rule specifies that for each flow $\mathbf{y}_f$ belonging to $\mathbf{Y}_g$, the value of attribute a for this particular flow $\mathbf{y}_f$ is equal to λ.

For example, in the case of an ICMP flooding attack, the vast majority of the associated flows use only ICMP packets, hence the absolute filtering rule {nICMP/nPkts = 1} makes sense, nICMP being the number of ICMP packets and nPkts being the number of packets.

4.3.2 Signature building

Once rules have been built, we intend to format them in order to ease the understanding by the network operator. In order to do so, we devise a method to rank relative rules. This section covers the signature building that uses as input the absolute and relative rules previously found.

4.3.2.1 Ranking relative rules

In order to construct a meaningful and simple-to-understand signature of the anomaly, we have to devise a procedure to rank and select the most discriminant filtering rules. Absolute rules are important, because they define inherent characteristics of the anomaly. As regards relatives rules, their relevance is directly tied to the degree of separation between normal and anomalous flows. We rank the degree of separation of anomalous traffic segments to the normal clusters using the well-known Fisher Score (FS), and select the top-K ranked rules. The FS measures the separation between clusters, relative to the total variance within each cluster. Given two clusters C_1 and C_2, the Fisher Score for feature a can be computed as:

$$F(a) = \frac{(\bar{x}_1(a) - \bar{x}_2(a))^2}{\sigma_1(a)^2 + \sigma_2(a)^2}$$

where $\bar{x}_d(a)$ and $\sigma_d(a)^2$ are the mean and variance of feature a in cluster C_d.

The more a traffic segment is far from the normal traffic, the more the behavior of its flows deviates from "normality". Furthermore, the more the behavior of anomalous flows is consistent inside a considered traffic segment, the more they are to be considered as dangerous. The same apply to normal flows. By taking into account variance of both normal and anomalous flows, the Fisher Score perfectly suits the relative rule ranking task.

4.3.2.2 Combining rules

To finally construct the signature, the absolute rules and the relative rules are merged into a single inclusive predicate, using the covering relation in case of overlapping rules. Signatures thus include both relative and absolute rules in a compact way.

4.4 Summary

This chapter addresses several techniques that we developed in order to process anomalies after their discovery. We first reduce the amount of extracted anomalies by correlating anomalies obtained from feature spaces built upon different aggregation levels l_i. We then reuse this result and two other simple anomaly characteristics to build a dangerousness index that allows us to rank anomalies. We finally present the characterization method that builds a detailed signature for each anomaly that tremendously help the operator in understanding the nature of the threat. This whole set of technique eases and reduces the work of the network operator.

These different techniques are further partially detailed in chapter 5 through a use case where UNADA is applied to a real network traffic trace.

Chapter 5

Evaluation

The evaluation of a network anomaly detection algorithm is a critical step to assess the relevance and efficiency of the algorithm design. It is the step that allows authors and readers to assess the interest of a proposed method by revealing its shortcomings and advantages. This chapter addresses this problematic.

In this chapter, we evaluate the ability of our algorithm to detect anomalies located in real traffic by using traffic traces from the public MAWI repository of the WIDE project [4]. The WIDE operational network provides interconnection between different research institutions in Japan, as well as connections to different commercial ISPs and universities in the U.S. The traffic repository consists of 15 minutes-long raw packet traces collected daily since 1999. The network traffic we shall work on consists of traffic from one of the trans-pacific links between Japan and the U.S. captured at two samplepoints named B and F.

In order to correctly assess our algorithm behavior, we run UNADA in an offline manner on the MAWI dataset. We first present an example of an execution of UNADA over a single time slot in order to help the reader to apprehend rationales and principles of our technique. We then carry out a more thorough analysis of our UNADA on subsets of the MAWI repository while using ground truth from the MAWILab dataset [83].

Unless specified otherwise, the UNADA parameters used in the next sections are the following ones : ws (or ΔT) $= 15s$, $epsDBS$ (DBSCAN$'$s parameter for neighborhood) $=$ $0.15, nbpDBS$ (DBSCAN$'$s parameter for neighborhood) $= 4, tICSim$ (cf; section 3.2.5.2) $=$ 0.85 and $tPropPtsNormC$ (or $Y, cf.\ section$ 3.2.6) $= 0.6$.

5.1 Single detailed case study: one time-slot in one trace in MAWI dataset

In order to ease the understanding of our algorithm for the reader, we expose the results of several processing steps of UNADA: from the combining step to the characterization step.

This section presents several results of an execution of UNADA over a single time slot. We use the MAWI trace on April 1 st 2004. This time slot lasts from the 405 th to the 420 th seconds in the trace. We execute UNADA as specified in section 4.1 in terms of aggregation levels used and with the parameter set specified in the introduction of chapter 5. We therefore only cluster flows aggregated through IP address source with /24 netmask (aggregation level l_3) and IP destination with /24 netmask (aggregation level l_6). We build a feature space based on the $A = 9$ attributes presented in Table 3.3 for each used aggregation level (l_3 and l_6) and then cluster the aggregated flows into two clustering ensembles $\mathbf{P}^{l_3}$ and $\mathbf{P}^{l_6}$.

We here present the UNADA steps after the subspace-clustering step. We first expose the results of the Intra-Clustering Results Association step. We then address the anomaly post-processing step with anomaly correlation, ranking and characterization.

5.1.1 Combination step with Inter-Clustering Results Association (ICRA)

This section presents the results of the combining step using ICRA. This step is applied over the clustering ensembles $\mathbf{P}^{l_3}$ (cluster ensemble obtained from aggregation according to source IP address /24, l_3) in section 5.1.1.1, and $\mathbf{P}^{l_6}$ (cluster ensemble obtained from aggregation according to destination IP address /24, l_6) in section 5.1.1.2 aggregations levels.

5.1.1.1 Combination step for clustering ensemble $\mathbf{P}^{l_3}$

$\mathbf{P}^{l_3}$ contains the cluster or outlier affiliation of each of the $F^{l_3} = 1015$ flows found using the l_3 aggregation level. Inter-CLuster Association (ICLA) phase results over $\mathbf{P}^{l_3}$ are displayed on figure 5.1. Each vertex is a cluster found in any of the generated sub-spaces. The graph contains 76 vertices. Since each vertex represents a cluster, this value is equal to the number of clusters found in $\mathbf{P}^{l_3}$ which is noted $p^{l_3}_{tot}$. It is therefore also equal to the size of the whole cluster set $\{C_0, C_1, .., C_{p^{l_3}_{tot}}\}$ within the clustering ensemble $\mathbf{P}^{l_3}$. Each vertex index thus has a value between 0 and $p^{l_3}_{tot} = 75$. Each edge means that the two linked clusters verify the similarity condition defined in equation 3.4. This condition states that the ratio of the cardinality of the intersection of the point set of the two clusters over the maximum of the cardinalities of the two clusters is greater than a threshold called tICSim. The cluster similarity threshold (tICSim) used is 0.85. This threshold value is used for all the graphs of the current section. It has been chosen from the result of the sensitivity analysis presented in section 5.2.3.3. The normal traffic is here circled and is represented by the vertex group with the highest number of vertices.

Figure 5.2 depicts the outlier similarity graph obtained over $\mathbf{P}^{l_3}$ during the Inter-Outlier Association (IOA) phase. The graph contains 79 vertices. Since each vertex represents an outlier, this value is equal to the number of outliers found in $\mathbf{P}^{l_3}$ which is noted $q^{l_3}_{tot}$. It is therefore also equal to the size of the whole outlier set $\{o_0, o_1, .., o_{q^{l_3}_{tot}}\}$ within the clustering ensemble $\mathbf{P}^{l_3}$. Each vertex index thus has a value between 0 and $q^{l_3}_{tot} = 78$.

5.1.1.2 Combination step for clustering ensemble $\mathbf{P}^{l_6}$

Corollary to the previous section, $\mathbf{P}^{l_6}$ contains the cluster or outlier affiliation of each of the $F^{l_6} = 1229$ flows found using the l_6 aggregation level. ICLA phase results over $\mathbf{P}^{l_6}$ are depicted on figure 5.3. Each vertex is a cluster found in any of the generated sub-spaces.

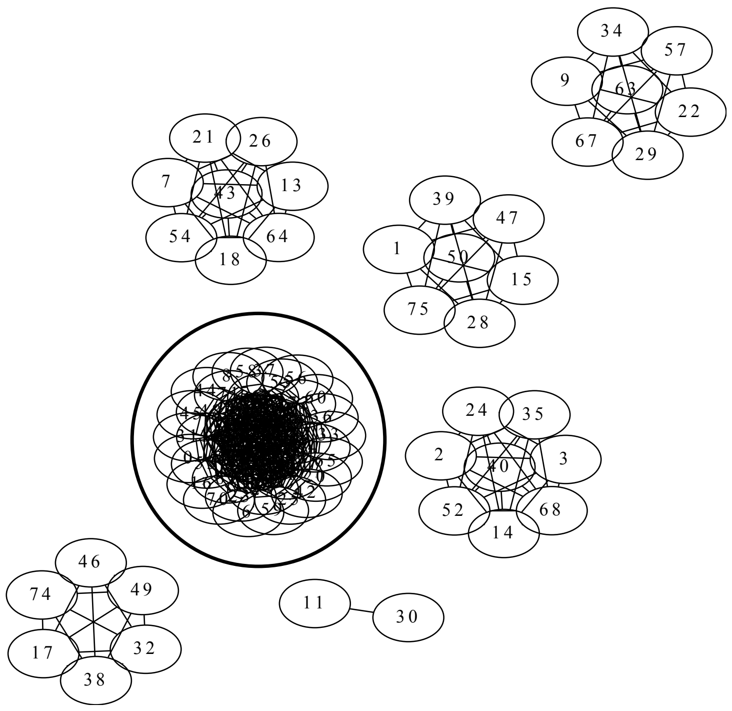


Figure 5.1: Cluster similarity graph for source address /24 aggregated data (aggregation level l_3).

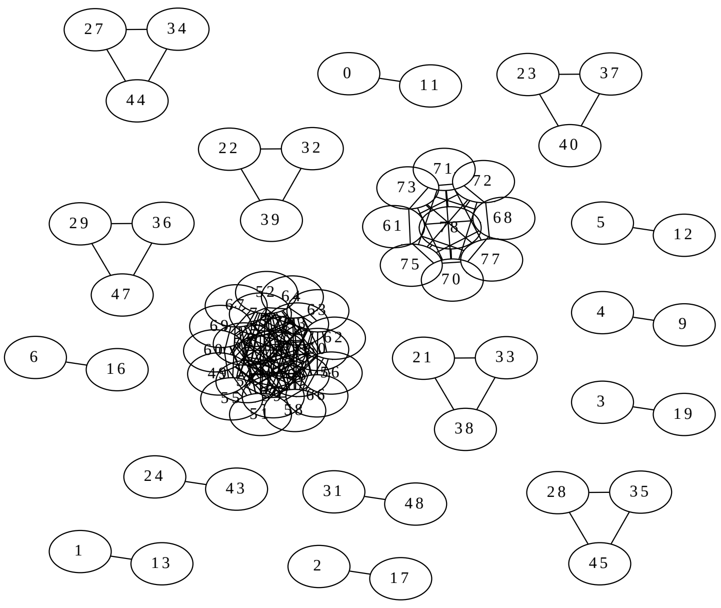


Figure 5.2: Outlier similarity graph for source address /24 aggregated data (aggregation level l_3).

The graph contains 77 vertices. Since each vertex represents a cluster, this value is equal to the number of clusters found in $\mathbf{P}^{l_3}$ which is noted $p_{tot}^{l_6}$. It is therefore also equal to the size of the whole cluster set $\{C_1, C_2, .., C_{p_{tot}^{l_6}}\}$ within the clustering ensemble $\mathbf{P}^{l_6}$. Each vertex index thus has a value between 0 and $p_{tot}^{l_6} = 76$. The normal traffic is here circled and is represented by the vertex group with the highest number of vertices.

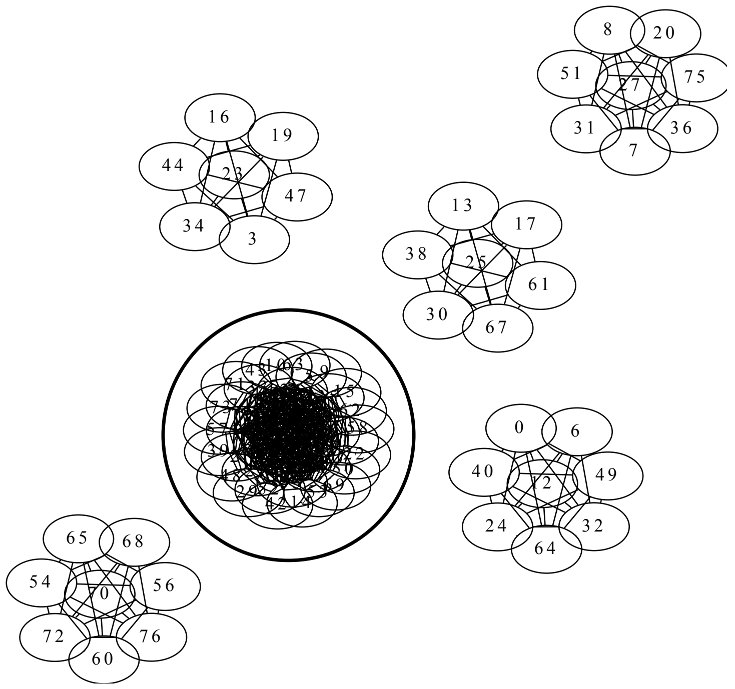


Figure 5.3: Cluster similarity graph for destination address /24 aggregated data (aggregation level l_6).

Figure 5.4 represent the outlier similarity graph obtained over $\mathbf{P}^{l_6}$ during the IOA phase. The graph contains 124 vertices. Since each vertex represents an outlier, this value is equal to the number of outliers found in $\mathbf{P}^{l_6}$ which is noted $q_{tot}^{l_6}$. It is therefore also equal to the size of the whole outlier set $\{o_1, o_2, .., o_{q_{tot}^{l_6}}\}$ within the clustering ensemble $\mathbf{P}^{l_6}$. Each vertex index thus has a value between 0 and $q_{tot}^{l_6} = 123$.

5.1.1.3 Similarity graph analysis

One can note that, in these four figures, every connected component is a clique. The hypothesis presented in section 3.2.5.2 (this hypothesis specifies that each vertex belongs to a single clique) is here completely verified.

Normal traffic can be identified in both cluster similarity graphs (cf. figure 5.1 and 5.3) as the circled vertex group. It is interesting to note that each of these two representations of normal traffic is a *single* clique that contains the biggest number of vertices. Anomalies are easily identified inside these graphs as small cliques. Inside outlier similarity graphs (cf. figures 5.2 and 5.4), every clique is an anomaly.

5.1.1.4 Traffic segments extraction

Once every graphs are built, here the four graphs depicted in figure 5.1, 5.2, 5.3 and 5.4, cliques are extracted and traffic segments are built upon each clique. Each traffic segment that contains less than 100-Y% of aggregated flows is considered as a potential anomaly. Each potential anomaly is assigned an index. In order to ease the comprehension of readers, we chose to assign one hundred anomaly indices for each of the four similarity graph

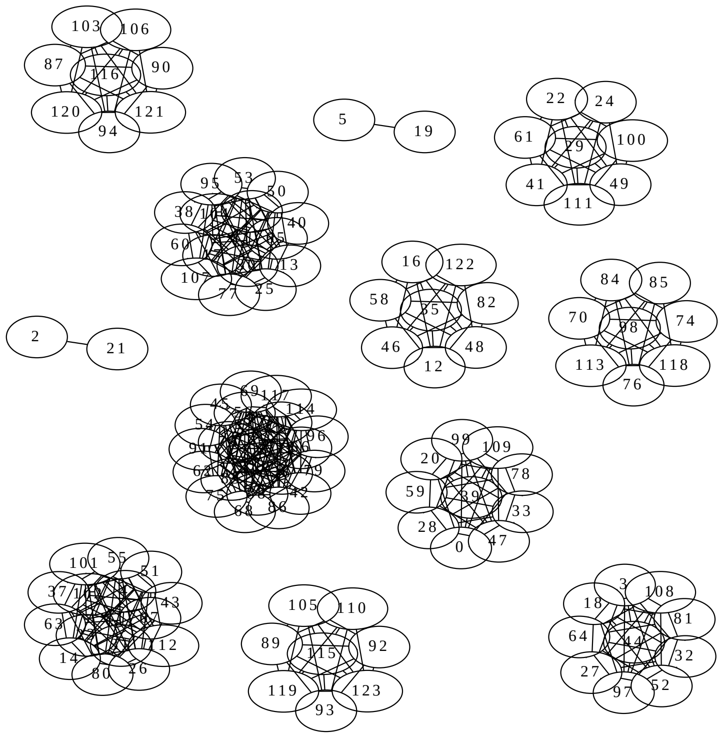


Figure 5.4: Outlier similarity graph for destination address /24 aggregated data (aggregation level l_6).

(cluster/outlier and source/destination). This will allow the reader to easily associate a traffic segment index located in a specific hundred to a specific similarity graph and thus a specific origin. This choice is made under the assumption that there is less than 100 traffic segments in each graph and can be easily overcome if needed. The anomaly index thus has a specific meaning. Values between 0 and 99 represent anomalies from clusters found in source IP address aggregated data (l_3). Values between 100 and 199 are associated anomalies from outliers found in source IP address aggregated data (l_3). Values between 200 and 299 index cluster found in destination IP address aggregated data (aggregation level l_6). Values between 300 and 399 are linked to anomalies from outliers found in destination IP address aggregated data (aggregation level l_6).

5.1.2 Anomaly post-processing

Once anomalies are extracted from feature spaces as traffic segments by the unsupervised detection, we apply the anomaly post-processing step. We here only present the anomaly correlation and characterization step. We do not show results of the anomaly dangerousness assessment step and only characterize correlated anomalies in order to not overload the reader and keep a concise and easy-to-understand outline.

5.1.2.1 Anomaly correlation

Anomaly correlation is employed in order to find flows that are different from the normal ones in both aggregation type: source IP address with netmask /24 (aggregation level l_3)

and destination IP address with netmask /24 (aggregation level l_6) . Figure 5.5 depicts the anomalies/traffic segments similarities as a graph. In this case, anomaly correlation extracts three edges from the graph. Each edge represents the link between two similar anomalies. Here, three correlated anomalies are present: 101 and 300, 111 and 305, and finally 100 and 205. The meaning of these indices are explained in section 5.1.1.4. First, the correlated anomaly between vertices 100 and 205 is a traffic segment built from outliers in l_3 (index 100) with a traffic segment extracted from clusters in l_6 (index 205). Second, the correlated anomaly between vertices 101 and 300 represents a traffic segment obtained from outliers in l_3 (index 101) with a traffic segment built upon outliers in l_6 (index 300). Finally, the correlated anomaly between vertices 111 and 305 is a traffic segment yielded from outliers in l_3 (index 111) with a traffic segment built from clusters in l_6 (index 305).

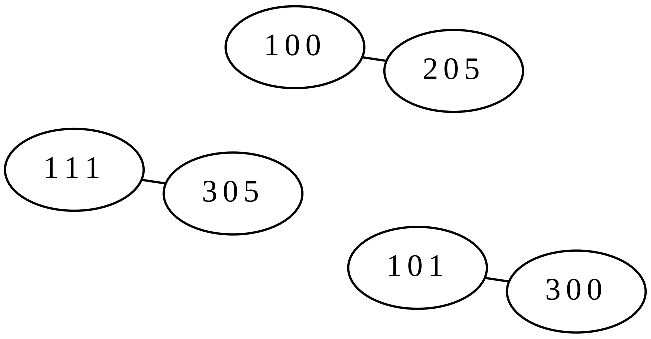


Figure 5.5: Anomaly similarity graph.

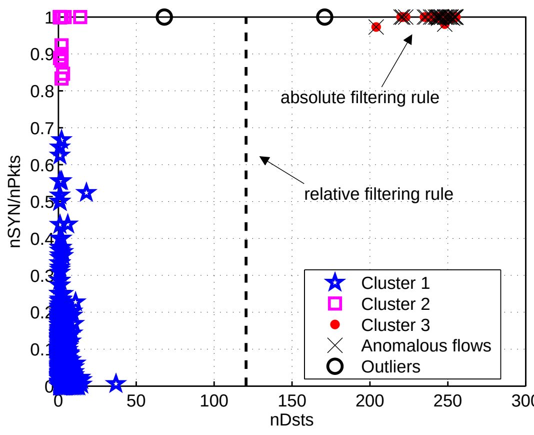
5.1.2.2 Anomaly characterization

We then apply characterization over these three correlated anomalies. Table 5.1 details each anomaly with its type, the traffic segment indexes extracted from ICLA and IOA and the two signatures detected from both source and destination aggregated data. The term "Few ICMP packets" actually means that these two anomalies were containing just a few harmless ICMP packets.

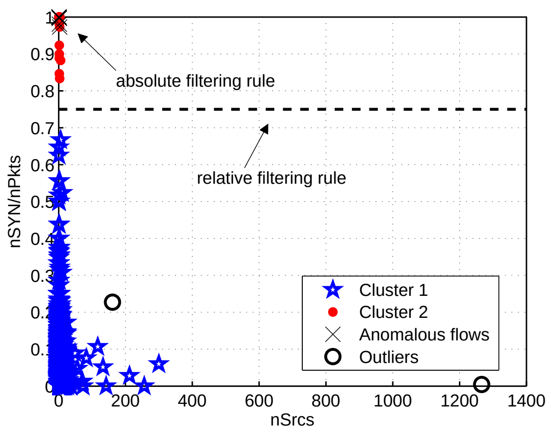
Anomaly type	Source traffic segment indice	Destination traffic segment indice	Source signature	Destination signature
Few ICMP packets	111	305	nSrcs = 1, nICMP/nPkts > λ_1	nSrcs = 1, nICMP/nPkts > λ_2
Few ICMP packets	112	309	nSrcs = 1, nICMP/nPkts > α_1	nSrcs = 1, nICMP/nPkts > α_2
Network scan	100	205	nSrcs = 1, nDsts > β_1, nSYN/nPkts > β_2	nSrcs = 1, nDsts > β_3, nSYN/nPkts > β_4

Table 5.1: Signatures of anomalies found.

Figures 5.6.(a,b) depict the results of the characterization phase for the network scan anomaly previously found. Each sub-figure represents a partition $P_n^{l_6}$ for which filtering rules were found. They involve the number of IP sources and destinations, and the fraction of SYN packets. Combining them produces a signature that can be expressed as (nSrcs $= 1$) $\wedge$ (nDsts $> \lambda_1$) $\wedge$ (nSYN/nPkts $> \lambda_2$), where λ_1 and λ_2 are two thresholds obtained by separating normal and anomalous clusters at half distance. In this particular case, we have $\lambda_1 = 120$ and $\lambda_2 = 0.75$. This signature makes perfect sense: the network scan uses SYN packets from a single attacking host to a large number of victims.



(a) SYN Network Scan (1/2)



(b) SYN Network Scan (2/2)

Figure 5.6: Filtering rules for characterization of the found network scan.

5.1.3 Summary

This section details several internal steps of the UNADA algorithm. It allows the reader to improve its comprehension of our algorithm. Several steps are detailed from the combining step inside unsupervised detection to the final characterization step.

5.2 Performance evaluation

This section addresses the performance evaluation of UNADA over real traces from the MAWI dataset. The evaluation of a network anomaly detection algorithm is a critical step that allows one to assess the performance of the proposed algorithm.

Our evaluation uses Receiver operating characteristic (ROC) curves as introduced by [91]. ROC curves are the standard statistical tool used to evaluate binary classifiers. ROC curves allow to visualize the trade-off between the True Positive Rate (TPR) and False Positive Rate (FPR) in a single curve. They are very interesting when few different classifier results are to be compared. They thus perfectly fit our needs. ROC curves' points are obtained by using several values for the threshold T_h applied to the dangerousness index processed at the anomaly ranking step (cf. section 4.2).

We first present the MAWILab dataset that provides ground-truth for the MAWI dataset in section 5.2.1. In section 5.2.2, we expose a first set of results on a limited subset of the MAWI dataset in order to provide preliminary results. In section 5.2.3, we perform a sensitivity analysis of our algorithm towards several of its parameters on a small subset of the WIDE repository. In section 5.2.4, we finally evaluate our tool over a wider subset of the WIDE repository.

5.2.1 MAWILab: a ground-truth for the MAWI dataset

The ground truth for our evaluation is provided by MAWILab [83]. In [83], Fontugne et al. combine four different anomaly detectors using the SCANN algorithm to produce a single set of anomalies. The used anomaly detection algorithm are: a PCA and sketch-based detector [97, 105], a detector relying on sketching and multi-resolution gamma modeling [92], a Hough Transform-based detector [143] and a detector relying on traffic feature distribution changes [93]. Their ground truth is a set of heuristics that allows one to label anomalous traffic. The output of their method is composed of four different labels: *anomalous*, *suspicious*, *notice* and *benign*. *Anomalous* means that the considered traffic is abnormal. Traffic labeled as *suspicious* is suspected to be abnormal and is situated above an internal threshold of SCANN. The *notice* label means that the traffic is below the same threshold. Traffic labeled as such has been identified as anomalous by one or several detectors but it shall not be considered as anomalous. *Benign* traffic has not been flagged as anomalous by any of the anomaly detection algorithms.

The MAWILab website [1] offers an access to the output of their method applied to MAWI traffic from samplepoints B (from December 2000 to June 2006) and F (from august 2006 to this day). The MAWILab output is itself divided in two files: one containing the traffic anomalies labeled as *anomalous* and *suspicious* and one containing the traffic anomalies labeled as *notice*. Each file is an XML file generated through the Admd library [2]. The Admd library provides a unified interface to export meta data for anomaly detection results as XML files. Anomalies are described as an event located in time and impacting a set of source and destination IP addresses, port and protocols.

The MAWILab authors also provide a tool called "MAWILab_Evaluate" that compares a reference XML file with another XML file containing the output of a detector to evaluate. MAWILab_Evaluate provides the classic statistical test results: true positive (TP), false positive (FP), false negative (FP) and true negative (TN). It provides these values for three different cases: flows, events and alarms. True positive flows represent the actual number of *anomalous* and *suspicious* IP flows identified as anomalous by a classifier. Statistical results on flows thus consider each flow atomically. On the contrary, true positive events represents anomalies for which at least one flow has been detected as anomalous. The number of true positive events thus represents a number of flows belonging to one or several anomalies provided that at least one flow of each anomaly has been detected as anomalous. The same applies for the other possible statistical test results, alarms. Statistical test results on alarms measure the number of alarms, and thus anomalies detected, provided that one flow of the considered anomaly has been detected. There are no true negative for this statistical test result. We therefore only use sensitivity or True Positive Rate (TPR) for alarms.

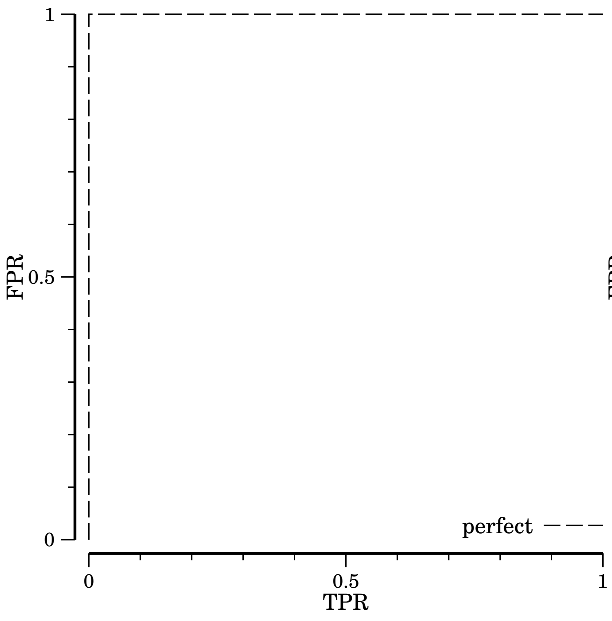
[1] http://www.fukuda-lab.org/mawilab/
[2] http://admd.sourceforge.net/

5.2.2 Preliminary results

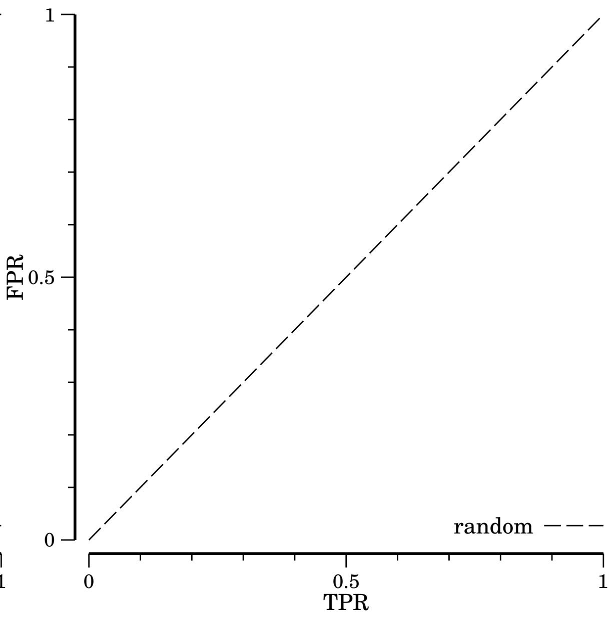
In order to expose the differences between events and flows, we first run UNADA on six MAWI traces: September 28th 2001, April 14th 2002, November 24th 2003, July 12th 2004, May 5th 2005 and October 13th 2006.

A perfect ROC curve would be shaped as a step, in other words, such curve would link the three following points $(0, 0)$, $(0, 1)$ and $(1, 1)$. Such ROC curve would offer a maximal TPR (i.e. TPR = 1) with a FPR equal to 0. The point at the top right corner represents the result where the TPR and the FPR are both equal to 1. Such result is achieved by labeling every flows as anomalous. Figure 5.7a display a perfect ROC curve.

On the other hand, the ROC curve obtained from a random classifier would be a diagonal, i.e. a curve that from $(0, 0)$ to $(1, 1)$. Figure 5.7b display such a curve.



(a) Perfect classifier ROC curve



(b) Random classifier ROC curve

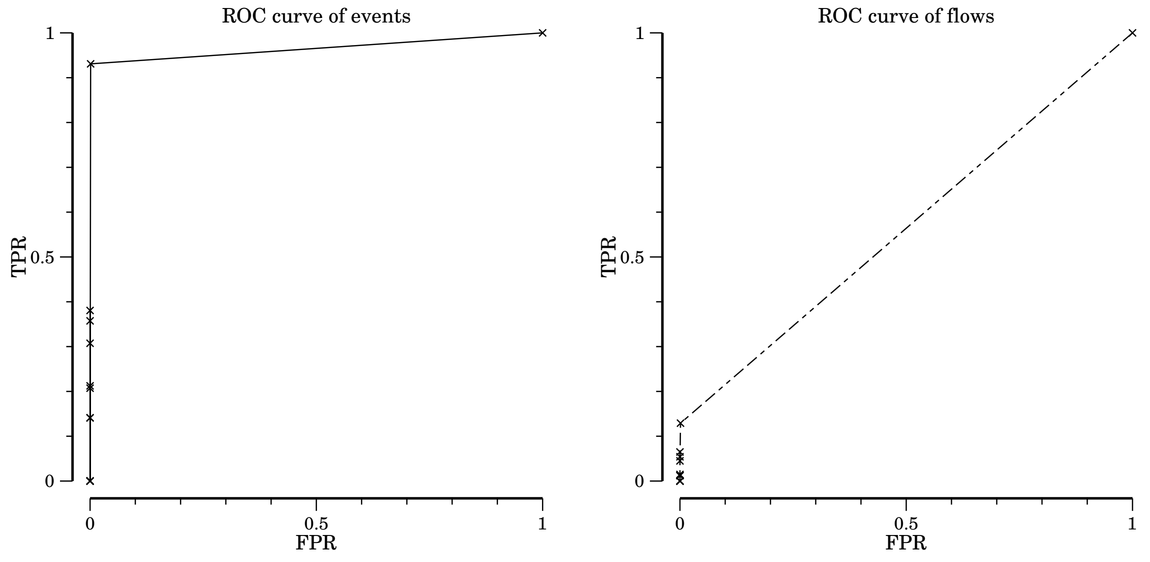
Figure 5.7: ROC curves for perfect and random classifiers.

In our case, UNADA always exhibits a very consistent low false positive rate. One can thus compare our ROC curve(s) to a leg: the point at $(0, 0)$ being the ankle and the point at $(1, 1)$ being the hip. The main inflexion is the knee. One can thus say that the more bended the knee, the closer the knee is to the point $(0, 1)$ and the better the results. UNADA's results can also be evaluated by measuring how high is the highest point located on the left part of the curve or, in the other words, how high and close to the top left corner is the knee. In fact, the FPR being always low, it is the position of this point that indicates how good are our performance in terms of TPR. This method is used throughout this section to assess the UNADA's performances.

Sensitivity curves display the value of sensitivity (or TPR) regarding the value of Dangerousness Index threshold. The shape of the sensitivity curve (the lower is the threshold, the high is the TPR) is easily explained by the fact that the lower is threshold, the more anomalies are reported. Since UNADA has a consistent low rate of false positive, high values of threshold offer poor interest because they exhibits low FPR and low TPR.

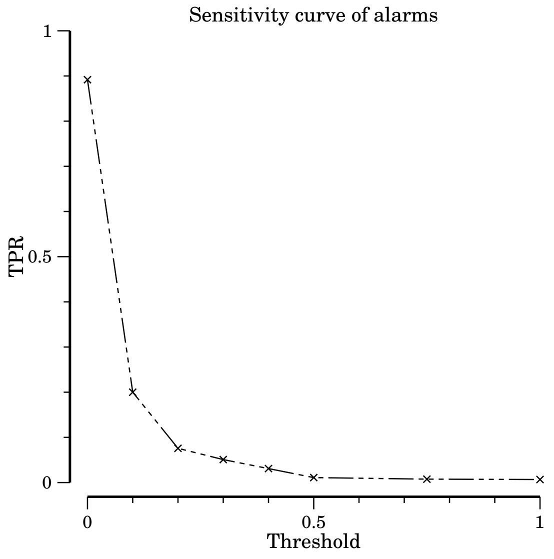
However, since the FPR is consistently low and since the lower the threshold, the higher the FPR, the main criterion to evaluate UNADA's performance is how high is the point with the lowest threshold value (i.e. the point located at the left of the sensitivity curve). Thus, the higher the point, the more anomalies are detected and the better the results.

Figures 5.8a and figure 5.8b expose the differences between events and flows. While MAWILab's events reflect the ability of an algorithm to find at least one flow in an anomaly, MAWILab's flows indicate the actual number of identified anomalous flows. Our algorithm here exhibits limited performance in identifying the whole sets of anomalous flows since it is only able to find 12% of anomalous flows. It however exhibits good performance in finding at least one flow in each anomaly by detecting 93% of anomalous events. It is also able to find 89% of the alarms.



(a) ROC curves of events

(b) ROC curves of flows



(c) Sensitivity curves of alarms

Figure 5.8: ROC curves for events and flows and sensitivity curve of alarms with change detection.

In order to understand why we have such performance in terms of overall detection of anomalous flows, we temporary remove the change detection step (cf. section 3.1.1) from UNADA and hence launch the clustering and post-processing step at each time slot. These results are displayed on figure 5.9.

The comparison between figures 5.8a and 5.9a, and figures 5.8c and 5.9c, on one hand, and figures 5.8b and 5.9b on the other hand, leads to several observations. The amount of detected events does not significantly increase since it only changes from 93% to 97%. The number of detected alarms increases from 89% to 98%. However, the number of detected flows nearly triples between figure 5.8b and 5.9b from 12% to 45%. This change is displayed on the curve by the fact that the knee of the curve is significantly higher in figure 5.9b than in figure 5.8b. This could be explained by the fact that some anomalies last more than one time slot and that IP addresses involved are not active during the whole duration of the anomaly. Thus, when we analyze every time slot, we are able to detect anomalies inside a greater number of time slots.

Another explanation that could explain the lack of performance on flow detection is the use of a relatively short time slot length of 15s. Anomalies that start at the end of a time slot and that finish at the beginning of the next one may be harder to extract. Their separation from normal traffic is indeed reduced by the fact that their packets are located in two time slots. Anomalies' attribute values are thus less extreme. We however did not investigate the impact of time slot size nor the use of sliding time window.

5.2.3 UNADA sensitivity analysis towards parameters

This section addresses the sensitivity of UNADA towards several of its parameters. Sensitivity analysis is critical in the sense that it allows one to verify that an algorithm is able to perform well when used with a wide range of settings. We here use a subset of the MAWI dataset generated by randomly selecting one trace for each month between January 2001 and December 2006. Contrary to the end of the previous section, we here use the change-detection step as described in chapter 3. This step allows us to obtain a prima facie evidence of an occurring anomaly.

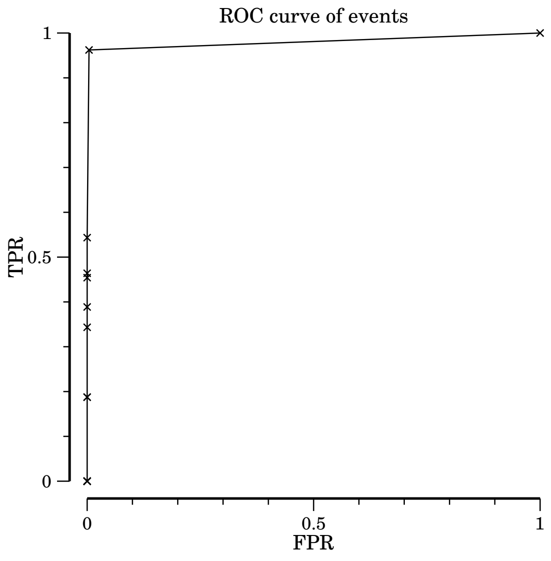
5.2.3.1 Time window size

We here study the influence of the time slot or time window size, noted ΔT or ws , over the performance of UNADA.

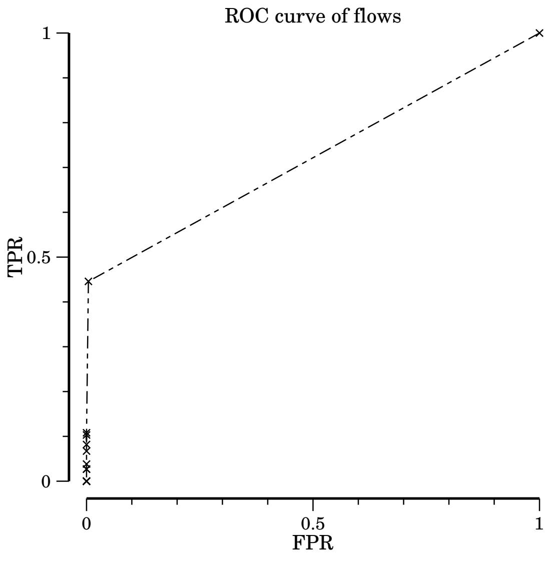
Figure 5.10a shows that the two highest knees of all curves are the ones of the curves for $ws = 15$ and $ws = 20$. $ws = 10$ and $ws = 30$ provide slightly degraded performance. $ws = 30$ provides the worst performance. Figure 5.10b show that $ws = 20$ and $ws = 30$ offer the best performance in terms of flows detection. $ws = 5$, $ws = 10$ and $ws = 15$ offer slightly lower performance. Concerning alarms, performances of all settings are close and follow this rule: the smaller the window the better the performance. The best overall performance are obtained by either $ws = 15$ or $ws = 20$.

5.2.3.2 Subspace clustering

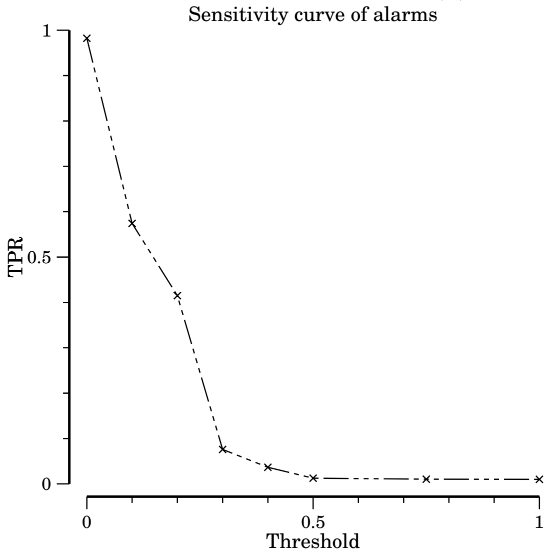
We here address the influence of the two parameters of the used clustering algorithm, DBSCAN [29], over our detection results. These parameters define the cluster density:



(a) ROC curves of events



(b) ROC curves of flows



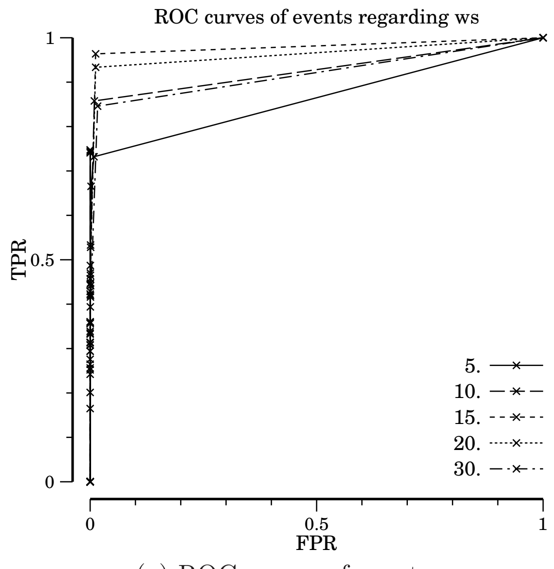
(c) Sensitivity curves of alarms

Figure 5.9: ROC curves of events and flows and sensitivity curve of alarms without change detection.

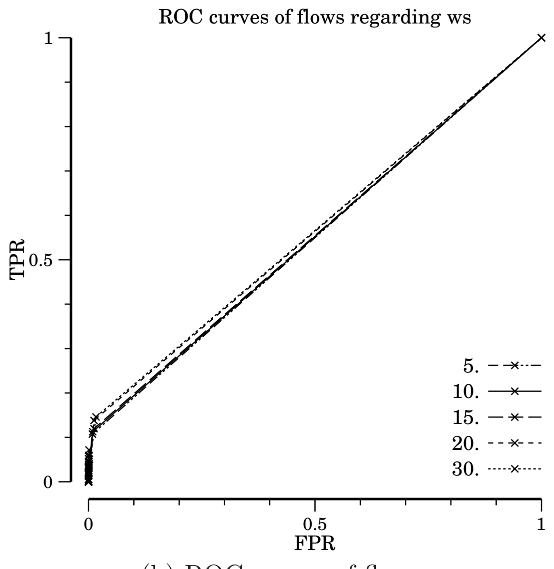
minimum number of points by cluster (nbpDBS) and epsilon (epsDBS).

Epsilon or ϵ defines the neighborhood of a point. In our case, the feature space is normalized for each attribute a and the dimension of each subspace $\mathbf{X}_n$ is 2. Epsilon is thus bounded between 0 and $\sqrt{2}$. When $\epsilon = 0$, each point is an outlier. On the contrary, since the feature space is normalized between 0 and 1 and the size k of each subspace is equal to 2, $\epsilon = \sqrt{2}$ guarantees us that each subspace contains a single cluster and no outliers.

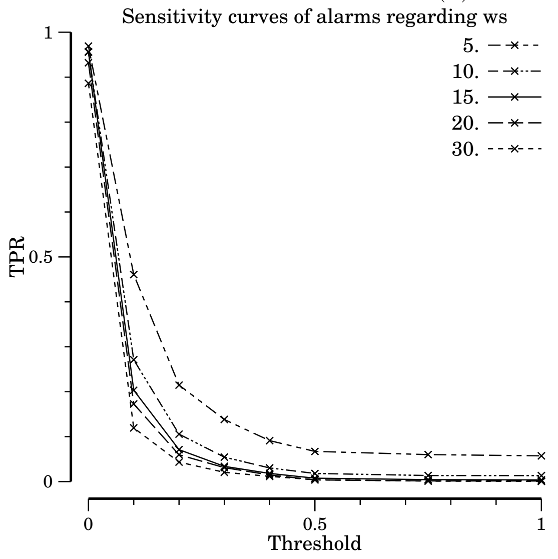
Figure 5.11a shows that the highest knee of all curves is the one of the curve for $epsDBS = 0.15$. $epsDBS = 0.15$ thus provides the best events detection performance. $epsDBS = 0.2$ exhibits a slightly lower knee compared to $epsDBS = 0.15$ and thus slightly degraded performance. $epsDBS = 0.1$ and $epsDBS = 0.3$ displays much worse



(a) ROC curves of events



(b) ROC curves of flows

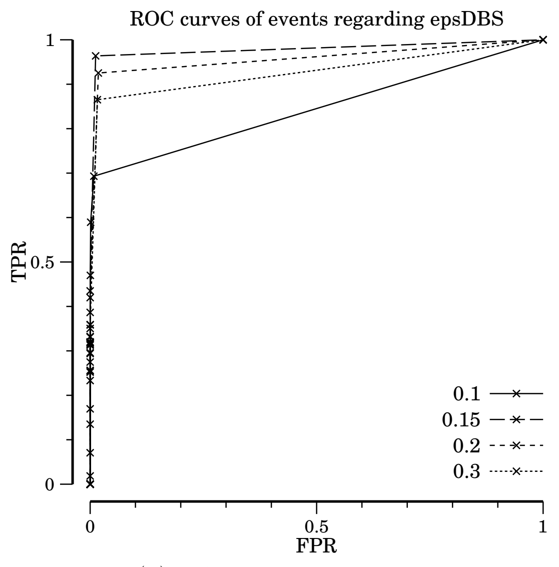


(c) Sensitivity curves of alarms

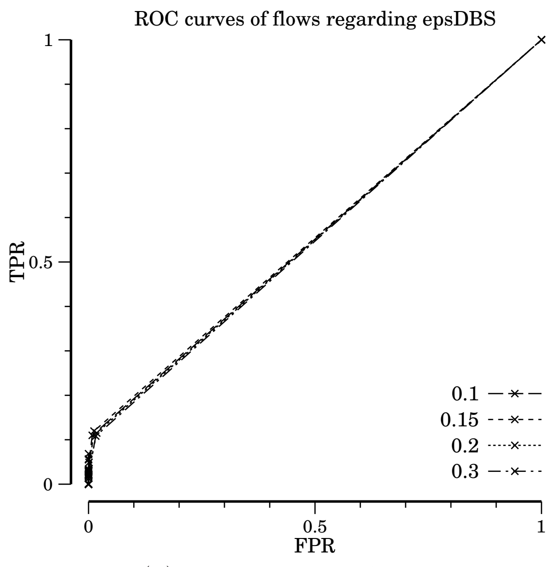
Figure 5.10: ROC curves for events and flows and sensitivity curve of alarms regarding time window size (ΔT/ws).

results since their knees are even lower. The observation of figure 5.11b does not allow one to differentiate any the parameter values in terms of flow detection. Figure 5.11c exhibits partially coherent results with the ones of figure 5.11a concerning $epsDBS = 0.15$ and $epsDBS = 0.3$. $epsDBS = 0.3$ displays the worst results in terms of alarms and $epsDBS = 0.15$ displays the best results. However, $epsDBS = 0.1$ here provides results in terms of alarms close to the best ones (i.e. $epsDBS = 0.15$). These results are better than the ones of $epsDBS = 0.2$. This was not the case for anomalous events. This could be explained by the fact that $epsDBS = 0.1$ and $epsDBS = 0.2$ detect different anomalies. $epsDBS = 0.1$ could be able to detect more anomalies than $epsDBS = 0.2$ but these anomalies could contain less flows.

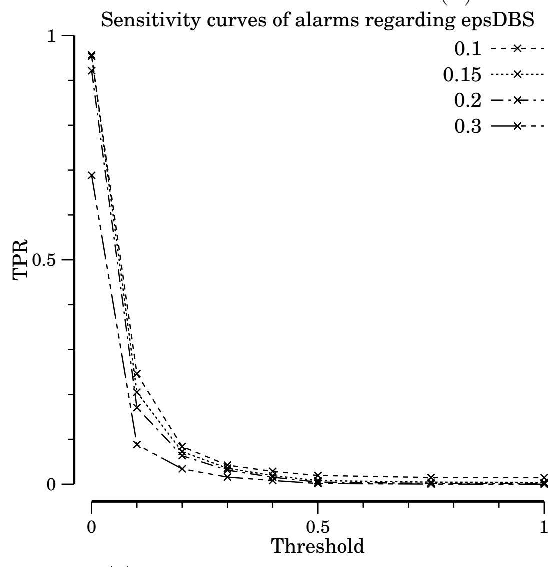
The minimum number of points by cluster (nbpDBS) represents the minimum num-



(a) ROC curves of events



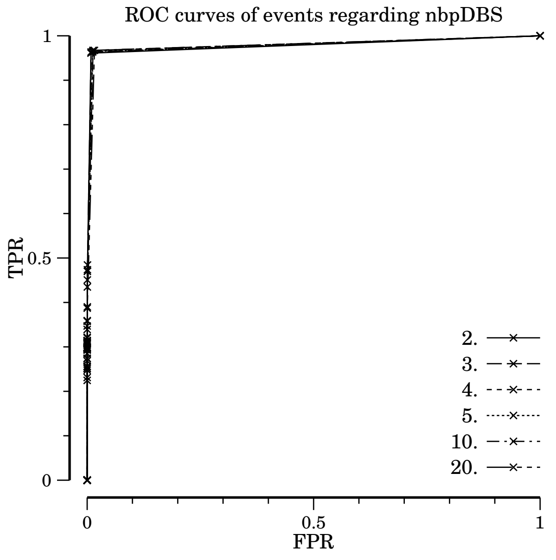
(b) ROC curves of flows



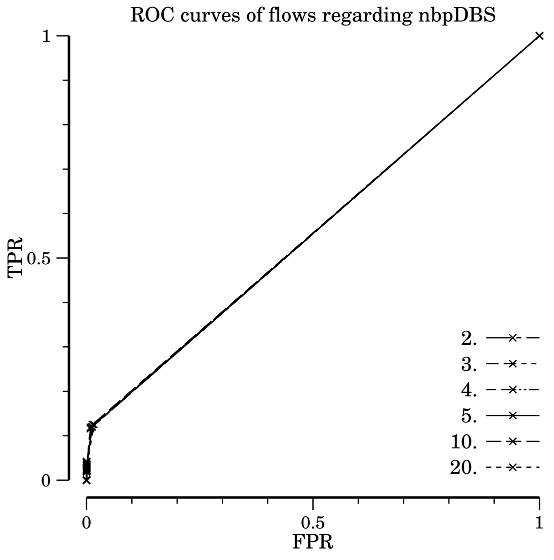
(c) Sensitivity curves of alarms

Figure 5.11: ROC curves for events and flows and sensitivity curve of alarms regarding epsilon (epsDBS).

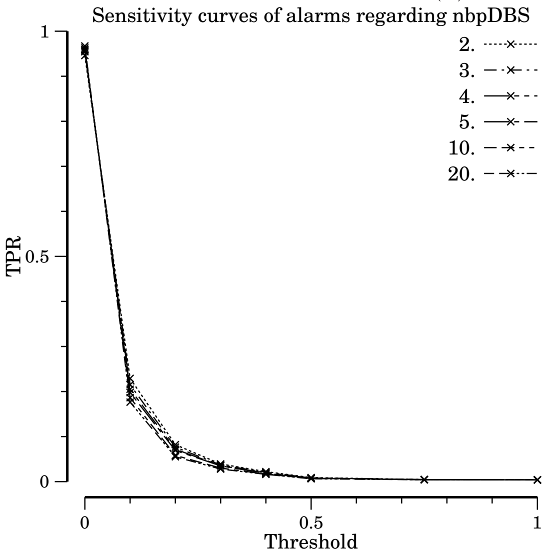
ber of points in the neighborhood of a considered point needed to form a cluster (this neighborhood being by epsDBS). This minimum number of points is bounded between 2 and the total number of points (aggregated flows in our case) inside the feature space. However, if we want to correctly find anomalous clusters, we shall use a small enough value in order to be able to find small cluster. We here chose to test values between 2 and 20. Figure 5.12 display the lack of influence of the minimum number of points by clusters on detected events, flows and alarms since all curves are close. There are several reasons that explained this behavior. First, the minimum number of points by cluster has no impact over the ability of UNADA to find a normal cluster in each subspace $\mathbf{X}_n$. Such impact would be possible if the chosen values for the minimum number of points by cluster would be high enough to cause the absence of a normal cluster as defined by the condition of hypothesis 3.2. The chosen values being bounded between 2 and 20, they never lead to the absence of a normal cluster since there are always at least one cluster that contains more than 20 points (or flows). Second, the considered parameter has no influence on whether an anomaly is considered as detected by MAWILab_Evaluate or not. Section 3.2.1 explains that anomalies can be represented either as clusters or as outliers. When the minimum number of points by clusters is too high, an anomaly does not contain enough flows to constitute a cluster (and thus a single traffic segment). In this case, the anomaly is represented by several outliers (and several traffic segments). But, from the point of view of MAWILab_Evaluate, both cases are identical since the same flows are tagged as anomalous. These two points explain the lack of influence of the minimum number of points by clusters parameter on detection performance.



(a) ROC curves of events



(b) ROC curves of flows



(c) Sensitivity curves of alarms

Figure 5.12: ROC curves for events and flows and sensitivity curve of alarms regarding the number of points by cluster (nbpDBS).

5.2.3.3 Inter-Clustering Results Association parameters

We here evaluate the influence of the cluster similarity threshold (tICSim) used in the ICRA step, and more precisely in the ICLA step (since such parameter is not needed for the IOA step, cf. section 3.2.5.2).

The cluster similarity (tICSim) used in ICLA is bounded between 0 and 1. 0 means that similar clusters may contain totally different aggregated flows. 1 means that the compared clusters must contain the exact same sets of aggregated flows.

Figure 5.13 shows that values $tICSim = 0.7$, $tICSim = 0.85$ and $tICSim = 0.95$ yield very similar performances since their curves are very close. This is especially true for figure 5.13b. In figure 5.13a, the highest point on the left part of the curve for $tICSim = 0.5$ is slightly lower than the one of the other settings. Corollary, figure 5.13c show that when $tICSim = 0.5$, UNADA detect slightly less correct alarms. $tICSim = 0.5$ thus exhibits a performance decrease and should therefore be avoided. Despite this slight degradation, the overall sensitivity of UNADA towards $tICSim$ is low.

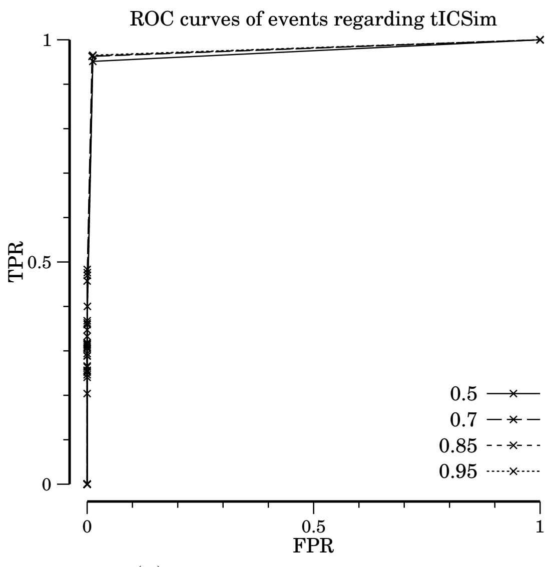
5.2.3.4 Normal traffic representation

We here examine the influence of the proportion Y of normal traffic in each subspace $\mathbf{X}_n$ and feature space (cf. hypothesis 3.2) over the performances of UNADA.

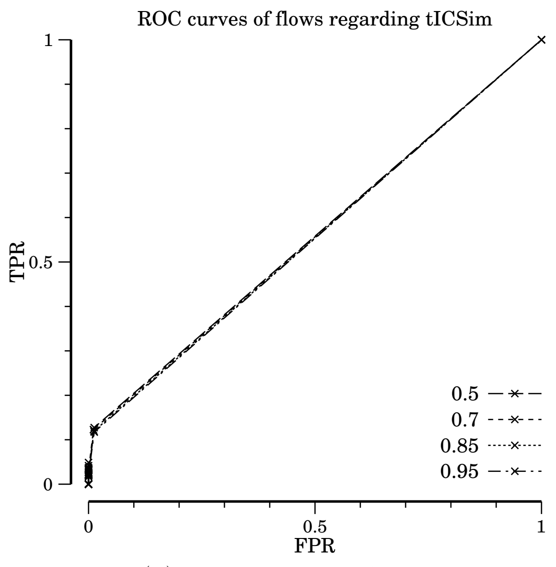
The proportion of flows in normal cluster/traffic segment (tPropPtsNormC) is bounded between 0 and 1. 0 means that, in each subspace, the biggest cluster will always be considered as the normal traffic (the same thing applies for the biggest traffic segment inside the feature space). 1 means that the biggest cluster must contain all aggregated flows in the feature space (the same thing also applies for the biggest traffic segment).

Figure 5.14 shows the impact of the proportion of flows in normal clusters/traffic segments (tPropPtsNormC) (also called the proportion Y of normal traffic in hypothesis 3.2), on events, flows and alarms. *tPropPtsNormC* values 0.5, 0.6, and 0.8 exhibit very similar and good performance since their ROC curves on both figures 5.14a and 5.14b are close. The analysis of figure 5.14c reveals the same behavior for these three parameter values. $tPropPtsNormC = 0.85$ or 0.9 however present less bended knees on ROC curves and a lower top-left point on the sensitivity curve. UNADA performance for these parameter values are thus deteriorated and they shall be avoided.

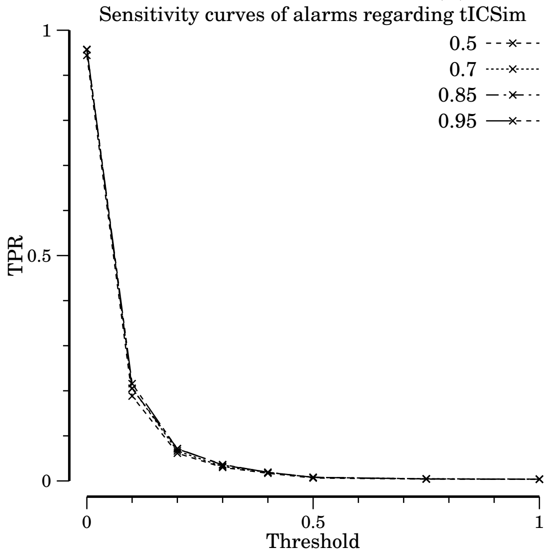
This deterioration can be easily explained by the fact that the threshold on the proportion of normal traffic, $tPropPtsNormC$, is used to enforce hypothesis 3.2 (and thus hypotheses 3.3 and 3.4). Therefore, when, for example, a cluster or a traffic segment contain 82% of the aggregated flows, $tPropPtsNormC = 0.85$ or 0.9 fails to find normal traffic while $tPropPtsNormC = 0.5$ or 0.6 or 0.8 manage to do so. In the former case, the execution stops and no anomaly is found in the current time slot while in the latter, UNADA continues its execution and find several anomalies. The impact of $tPropPtsNormC$ is thus critical over the ability of UNADA to find a representation of the normal traffic inside the feature space. It is however interesting to note that the range of appropriate values for $tPropPtsNormC$ is relatively wide. UNADA thus seems to be easily tunable regarding this parameter.



(a) ROC curves of events



(b) ROC curves of flows



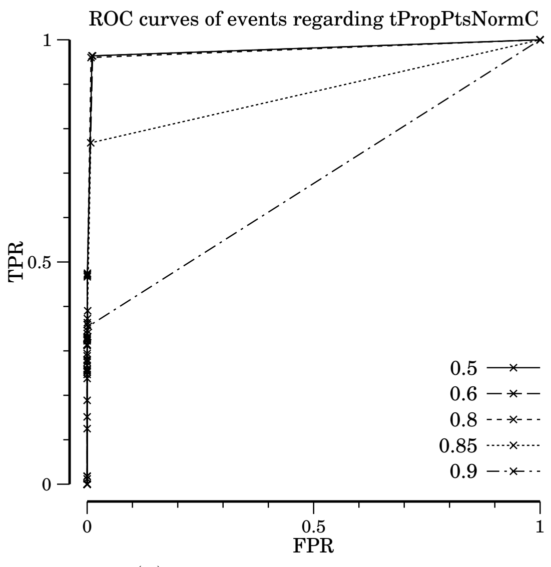
(c) Sensitivity curves of alarms

Figure 5.13: ROC curves for events and flows and sensitivity curve of alarms regarding the cluster similarity (tICSim).

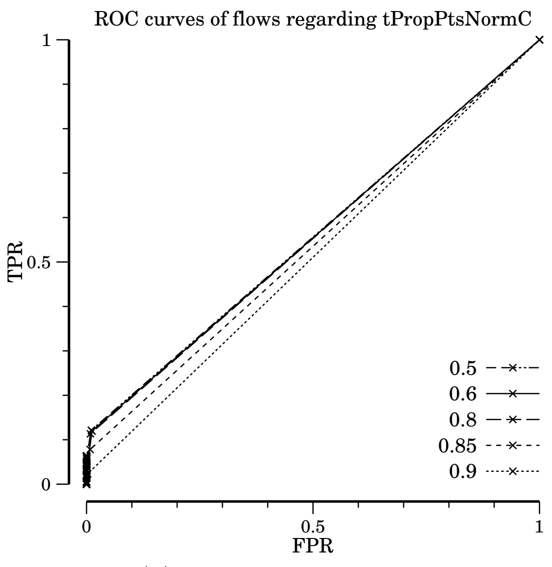
5.2.4 Global evaluation on the MAWI dataset

In order to have a more thorough and reliable view of the global results of UNADA on the MAWI dataset, we increase the number of traces randomly selected from the MAWI dataset. We thus randomly pick four traces for each month from January 2001 to December 2006 (instead of one for each month in section 5.2). The UNADA parameters are those which exhibit the best performance in the sensitivity analysis section: ws (or ΔT) $= 15s$, $epsDBS = 0.15$, $nbpDBS = 4$, $tICSim = 0.85$ and $tPropPtsNormC$ (or Y) $= 0.6$.

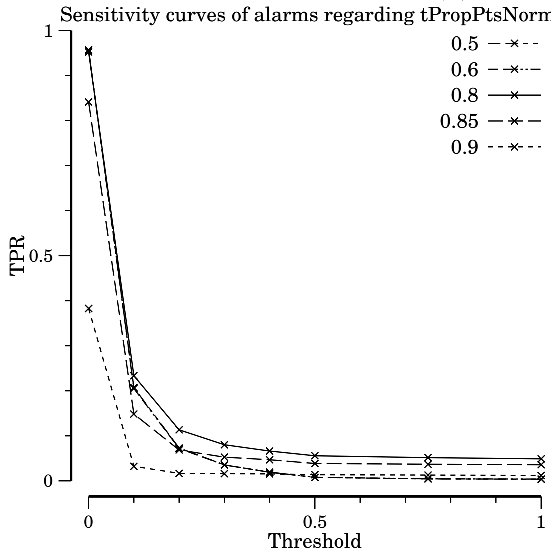
The ROC curves show that UNADA is able to detect more than 95% of anomalous events documented by MAWILab. Our algorithm however exhibits limited performance in terms of anomalous flow detection. UNADA also displays a constant and very low rate of false positive across statistical tests (events and flows). This is especially important



(a) ROC curves of events



(b) ROC curves of flows

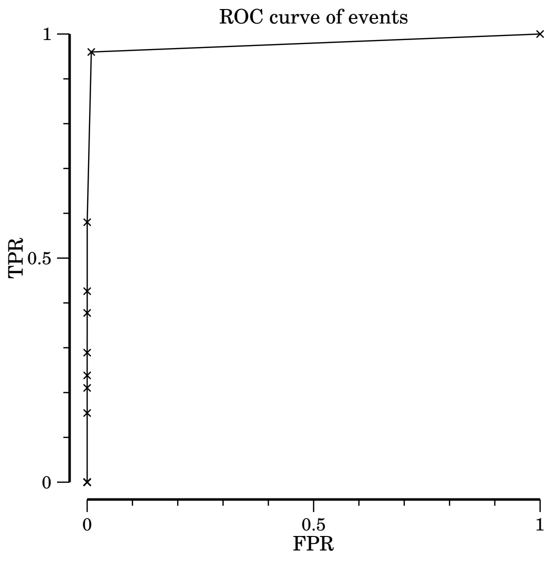


(c) Sensitivity curves of alarms

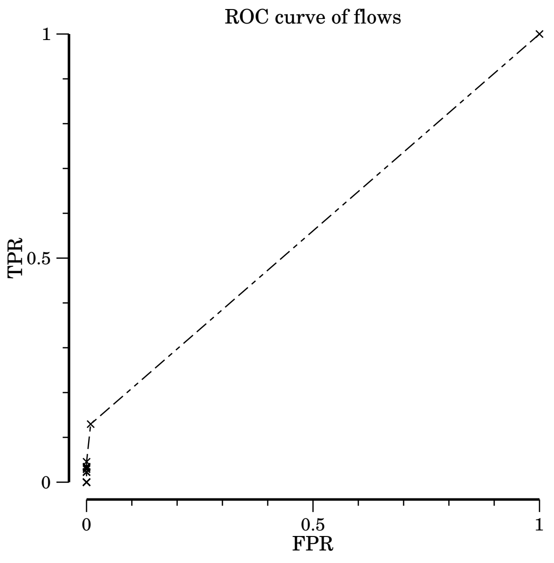
Figure 5.14: ROC curves for events and flows and sensitivity curve of alarms regarding the proportion of flows in normal cluster/traffic segment (tPropPtsNormC).

since it reduces the risk of alarms saturation for the operator.

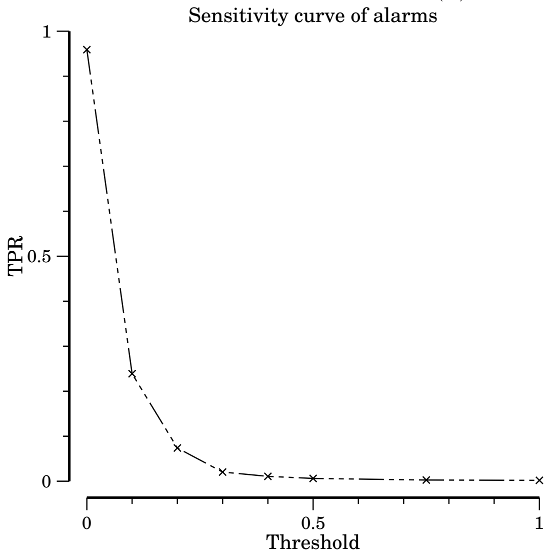
We then proceed with an indirect comparison with the detectors used in MAWILab. In [83], Fontugne et al. states that the detector that detected the most anomalies in their study is the Kullback-Leibler-based [93]. This detector contributed to half of the anomalies reported in MAWILab. Since UNADA is able to detect at least one flow in more than 50% of the anomalies of MAWILab, UNADA has better performance than any detector used in MAWILab in terms of number of detected anomalous events. This thus demonstrate the very good versatility of UNADA: it can detect a wide range anomalies and obtain results close to the ones obtained in MAWILab through a strategy of anomaly detectors results combination.



(a) ROC curves of events



(b) ROC curves of flows



(c) Sensitivity curves of alarms

Figure 5.15: ROC curves for events and flows and sensitivity curve of alarms from 2001 to 2006 with 4 traces by month.

5.3 Summary

This chapter addresses the evaluation of UNADA. UNADA's performance is measured against ground truth provided by the MAWILab work. We evaluate UNADA's parameters' influence over the performance of our algorithm on a subset of the MAWI dataset. UNADA exhibits a low sensitivity towards two of its parameters: *nbpDBS* and *tICSim*. UNADA's sensitivity towards the parameters *ws*, *epsDBS* and *tPropPtsNormC* is greater. However, the range of appropriate values for these parameters is wide enough to offer an easy tuning.

Our algorithm exhibits a consistent very low rate of false positive. This is very interesting because it guarantees that the network operator will not be overwhelmed by false alarms. UNADA suffers from limited performance on anomalous flows identification.

There are several factors that explain this lack of performance: first, we do not launch the unsupervised detection at every time slot which lead us to detect only parts of long lasting anomalies and second, we use consecutive time slots that cause short anomalous flows to be invisible from UNADA point of view. Finally, UNADA presents consistent and very good performances in terms of alarms and events detection. It is thus able to detected more than 95% of anomalous events on a substantial subset of the MAWI dataset. This ensures us that at least one flow is detected inside the vast majority of the anomalies documented by MAWILab. As a consequence, our algorithm only misses very few anomalies.

Chapter 6

Conclusion

Network traffic changes. These changes can be seen through several perspectives. A first perspective is spatial: traffic changes according to where one performs measurements. The second possible perspective is temporal. Network traffic changes in two ways along time. On one hand, traffic displays a constant volume increase and cyclic variations. On the other hand, traffic nature evolves: new applications arise, macro-economic and politic changes occur, etc. Network anomalies follow a similar path: they change along space and time. Network anomaly detection systems shall therefore seamlessly cope with these changes. Operators need to rely on tools that can autonomously cope with this novelty. We aim at designing a tool that would fulfill this goal by using as little knowledge about traffic as possible.

This chapter summarizes and comments the contributions of our work. We address each contributions previously presented and expose its advantages and shortcomings.

6.1 Contributions

This section addresses our two major contributions that constitute our tool UNADA, unsupervised network anomaly detection and our anomaly post-processing techniques.

6.1.1 Unsupervised network anomaly detection

The goal of UNADA is to find anomalies inside network traffic using as little knowledge about traffic as possible. In order to do so, our algorithm only relies on two hypotheses: anomalous traffic is statistically different from normal traffic and normal traffic represents the majority of traffic.

We first propose a representation of network anomalies according to the flows aggregation method used: either source or destination IP addresses, both with several netmasks. This representation allows us to model normal traffic and anomalies inside a feature space constituted by several attributes built on each aggregated flow.

We propose a clustering method based on density-based clustering and the notion of clustering ensemble. This method combines the clustering results in low-dimensional spaces to form a global partition of the feature space. This method is highly reliable to find abnormal outliers and both abnormal and normal clusters since it allows UNADA to detect the majority of the anomalous events present in MAWI and documented by MAWILab (cf. section 5.2.4).

We also show that clustering is the most time-consuming task in our current implementation. We therefore analyze the scalability of our approach and show that it scales reasonably well regarding the number of flows and number of features or attributes used. We state that, since the clustering step in UNADA can be parallelized, it can be used in real-world provided that enough resources are available.

6.1.2 Anomaly post-processing

In order to process found anomalies, we propose a corpus of techniques. First, we present an anomaly correlation step that aims at grouping similar anomalous traffic segments found in several feature space to reduce redundancy among alarms. We also propose an anomaly ranking technique that reuse the results of the anomaly correlation step and use two other anomaly characteristics (proportion of traffic belonging to this anomaly in terms of number of packets and bytes) to rank anomalies according to their dangerousness. We finally present a method that characterizes anomalies through filtering rules that are further combined to produce compact and easy-to-understand signatures.

6.1.3 Evaluation

Our work is evaluated over a subset of the MAWI dataset using the ground truth provided by the MAWILab dataset. UNADA exhibits limited performances concerning the overall detection of anomalous IP flows and a very good ability to detect anomalous events. In other words, UNADA is only able to detect a small fraction of the flow set of each anomaly but is also able to find at least one flow in the majority of anomalies. We show that one of the reasons of this is the fact that the unsupervised detection and characterization is not launched at each time slot (cf. 5.2.2).

6.2 Prospectives

We here present the prospectives concerning several UNADA's aspects that ought to be improved and extended in the future.

6.2.1 Hypotheses on traffic nature

UNADA uses hypotheses 1.1 and 3.2 to separate normal traffic from anomalous one. Hypothesis 1.1 is widely used among network anomaly detection systems in literature and is also widely accepted by the community. On the other hand hypothesis 3.2 comes from the intrusion detection field and has never been used for network anomaly detection up to our knowledge. It is a very restrictive hypothesis since it forbids the existence of both several mid-size normal cluster inside each feature-space and several mid-size normal traffic segments in the whole feature space. The evaluation carried out in section 3.2.5.2 however shows that, despite the use of this very constraining hypothesis, UNADA performs very well. This can be considered as an "a posteriori" validation of the hypothesis validity.

If one chooses to not use this hypothesis, the problem of identification of normal traffic remains. This problem seems very hard to solve since one has to devise a method to separate normal clusters from anomalous ones. Such heuristics may be based upon thresholds on the size and number of clusters. However, they may appear as very difficult to properly tune when used in real world.

6.2.2 Clustering technique

We think that future work shall aim at integrating state-of-the-art high-dimensional data clustering algorithms similar to those presented in [30] and evaluated in [74] and [77]. Some of the algorithms evaluated in such work seem very promising regarding their scalability towards the number of points (in our case aggregated flows) and attributes used. Such algorithm would allow us to use a much bigger amount of aggregated flows and a much wider set of attributes. This would improve both the unsupervised detection step, by easing the separation from normal traffic, and the characterization step, by providing a greater number of attributes to build rules on. This may also allow us to use unsupervised learning at each time slot and at a reduced cost thus improving our performance in terms of number of anomalous flows detected (cf. section 5.2.2). A multi-resolution time-slot scheme would also be less expensive to use.

The use of such clustering algorithms and of new attributes may require the reexamination and adaptation of the hypothesis on the proportion of normal traffic according to the behavior of the considered clustering algorithm.

6.2.3 Anomaly post-processing

This section present prospectives on our three anomaly post-processing methods: anomaly correlation, anomaly ranking and anomaly characterization.

6.2.3.1 Anomaly correlation

Prospectives about anomaly correlation in terms of functionality rely in the extension to clustering results built upon aggregation levels different than the two aggregation levels presented in 4.1: IP source/24 (l_3) and IP destination/24and l_6). Clustering results built on all the aggregation levels presented in section 3.1.2: $l_{1,2,3}$ (IPsrc/8, /16, /24) and $l_{4,5,6}$ (IPdst/8, /16, /24) could for example be correlated between each other. One can also imagine to create new aggregation levels based on port, type of port (less than 1024 and over 1024).

In terms of evaluation, anomaly correlation is partially analyzed through a single example in section 5.1. A more thorough evaluation could have been lead if we had been able to automatically analyze the anomaly groupings done by our anomaly correlation method. However, this is not possible since MAWILab_Evaluate cannot compare the actual inclusion between reference anomalies and anomalies found by the tool to evaluate. Furthermore and from a more general point of view, partial manual analysis of the global results reveals that the alarms generated by our tool are different than the ones in the MAWILab dataset. This means that there is a high probability that alarms from either systems are included in the alarms of the other system. Unfortunately, MAWILab does not provides such information yet. Future improvements of our anomaly correlation method and its evaluation shall therefore first find a way to obtain this information.

6.2.3.2 Anomaly ranking

Future work shall intend to improve anomaly ranking by adding an assessment of the separation of anomalous traffic representations from normal traffic in the feature space to the formula of the dangerousness index. Such separation could for example reuse the Fisher Score built upon traffic segments for ranking of relative rules. The rationale would be to assess a greater anomalousness for anomalous traffic segments far from normal traffic.

6.2.3.3 Anomaly characterization

Future work in terms of anomaly characterization relies on improvement of the filtering rule building method. Up to now, it uses very simple heuristics based on two types of rules: relative and absolute rules. A future work, could reformalize the building of these rules through a supervised learning-based approach. In fact, since hypothesis 1.1 and 3.2 allow us to identify normal traffic, we are able to provide labels for normal and anomalous traffic. A supervised learning technique could then be applied to automatically find rules that separate anomalous flows from normal ones. Decision tree algorithm such as ID3 [144] or C4.5 [145] would perfectly fit this task.

Section 5.1 provides an insight of the result of our characterization method. However, we have not been able to experimentally and thoroughly assess the efficiency of our characterization step. Such assessment could have been realized by comparing our generated signatures with signatures that match real anomalies detected by signature-based systems. The MAWILab repository however does not provide classification of the detected anomalies. Its XML-based anomaly annotation method already provides a field called type. This field is however used to store the four different types of output of MAWILab: anomalous, suspicious, notice and benign. A new field containing anomaly nature could easily be added. We thus let the characterization method evaluation to future work.

6.3 Summary

UNADA constitutes an original and new application of unsupervised machine learning in the field of network anomaly detection. We use clustering techniques to extract anomalies from traffic while only relying on two hypotheses, anomalies statistically deviates from normal traffic and they represent less than half of the traffic. We thus build a system that autonomously cope with traffic changes. We also propose a set a techniques to automatically and autonomously correlate, rank and characterize anomalies that process found anomalies. We verify that our algorithm is able to find anomalies against documented network traces from the MAWI repository.

In the context of our work, we also want to emphasize on the interest of datasets that provides ground-truth for networking problems (network anomalies with MAWILab

[1], traffic classification through the UNIBS dataset [2], etc.). These dataset are critical to automatically and easily evaluate work, especially when the considered work include machine learning techniques.

We apprehend our work as a significant proof of the interest of the use of machine learning in networking. Our work especially emphasizes the interest of unsupervised learning application to cope with use cases that face changing environment and need to autonomously adapt themselves.

[1]http://www.fukuda-lab.org/mawilab/

[2]http://www.ing.unibs.it/ntw/tools/traces/

Détection non supervisée d'anomalies du trafic réseau

Johan Mazel

26 janvier 2012

Résumé en français

Chapitre A

Introduction

La détection d'anomalies dans l'Internet actuel est une tâche aussi compliquée que fastidieuse. Les attaques de déni de service (Denial of Service, DoS), de déni de service distribué (Distributed Denial of Service, DDoS) et les scans menacent l'intégrité et le fonctionnement quotidien du réseau. La principale difficulté liée à la détection et à l'analyse de ces différentes anomalies dans les réseaux est leur capacité à évoluer et à s'amplifier [?].

Le principal problème des systèmes de détection d'anomalies réseaux est que ces anomalies évoluent. Il est ainsi impossible de définir un ensemble d'anomalies existantes. De nouvelles anomalies apparaissent chaque jour. Ce problème est accentué par le fait que le trafic de fond, légitime, évolue lui-aussi dans le temps.

Ce travail de thèse vise à trouver concevoir un système de détection d'anomalie du réseau capable de détecter une vaste palette d'anomalies et utilisant le moins possible d'information. Le reste de ce chapitre est composé de deux sections. La première de ces sections expose les raisons qui nous amènent à penser que les systèmes de détection d'anomalies doivent être plus autonomes et comment nous atteindrons ce but. La seconde section détaille nos contributions.

A.1 Problèmes initiaux

A.1.1 Trafic changeant dans l'Internet

A.1.1.1 Trafic changeant

Le trafic réseau change. Ce changement s'opère suivant 3 axes que nous détaillons ci-après. Le premier de ces axes est le changement du trafic dans le temps, suivant plusieurs échelles et à travers plusieurs métriques comme le nombre de paquet ou d'octets. Nous appelons ce changement l'évolution du trafic. Ces changements d'ordre temporels sont de deux types : cycliques selon plusieurs échelles (journalières, hebdomadaires, etc) ou cohérentes en termes de tendances dans le temps : constante augmentation du traffic. Les variabilités journalières et hebdomadaires ont par exemple été mises en évidence dans [1] et [2]. Dans [2], Papagiannaki et al. observent que le trafic dans 2 des 3 Points de Présence (PdP) augmente entre Octobre 2000 et Juillet 2002. Dans [5], Cho et al. montrent que le trafic dans les principaux points d'échange Internet du Japon a constamment augmenté au cours des dix dernières années.

Le second axe de changement est la disparité. Ce terme recouvre les différences de structure du trafic lorsqu'on l'observe depuis différents points de vues. Ce phénomène est mis en évidence par plusieurs travaux. Dans [7], Ringberg et al. exposent la différence du nombre moyen de paquets par flux selon la méthode d'agrégation du trafic : par liens, par routeurs d'entrée ou par flux origine-destination. Dans [2], Papagiannaki et al. montrent que l'évolution du trafic est différente sur les trois PdP observés.

Le troisième et dernier axe de changement du trafic est son changement de nature. Nous appelons ce phénomène la mutation du trafic. Un premier exemple de ce phénomène est la soudaine apparition du service Netflix. Le cas particulier de Netflix au Canada est un bon exemple. Netflix y a été lancé le 22 septembre 2010. Ce service représentait, fin mars 2011, 13.5% du trafic descendant aux heures de pointes [9].

A.1.1.2 Anomalies changeantes

Les anomalies connues sont nombreuses de nature très différentes. Certaines anomalies ne sont pas dangereuses en tant que telles mais peuvent être vues comme des indices d'une attaque future. Les balayages de port entrent dans cette catégorie. Les anomalies visant le déni de service comme les *SYN flood* ou *ICMP flood* sont dangereuses dans la mesure où elles cherchent à dégrader le service offert par leur cible.

Dans [12], Borgant et al. réalisent une étude longitudinale de la base de traces du projet WIDE [4]. Les auteurs montrent que certaines anomalies comme les *SYN flood* sont récurrentes alors que d'autres anomalies sont plus localisées dans le temps. Dans [16], Allman et al. analysent le trafic entrant du réseau du LBNL et plus particulièrement les balayage de ports du 1 ^{er} Juin 1994 au 23 Décembre 2006. Les auteurs montrent que les balayages de ports évoluent tout au long de leur étude. Ceux-ci augmentent lorsque de nouveaux vers apparaissent. Les numéros de ports, et donc les services visés, évoluent aussi.

A.1.2 Pourquoi les systèmes de détection d'anomalies du trafic actuels sont déficients ?

Deux approches existent dans la littérature et dans les outils existants : la détection par signature (*misuse detection*) et la détection d'anomalie (*anomaly detection*). Les systèmes basés sur des signatures sont très efficaces pour détecter les anomalies dont les signatures sont connues. Cependant, ils ne peuvent défendre les réseaux contre des anomalies inconnues. De plus, la construction et la mise à jour des signatures sont coûteuses et fastidieuses car nécessitant une intervention humaine.

Les systèmes de détection d'anomalie du trafic basés sur la détection d'anomalie utilisent un trafic dépourvu d'anomalie afin de construire un modèle de trafic normal [17, 18]. Ce modèle permet alors d'isoler les anomalies en tant qu'instances déviant du modèle précédemment construit. Ce type de détection est capable de mettre en évidence de nouvelles anomalies. Cependant, la construction du profil de trafic normal peut être compliquée et elle suppose que l'on ait accès à du trafic dépourvu d'anomalie. De plus, l'évolution continue des caractéristiques du trafic complique la mise à jour du profil de trafic normal.

L'apprentissage supervisé a parfois été utilisé afin d'automatiser la construction des signatures [20] et du profil de trafic normal [19]. Les méthodes présentées dans ces travaux permettent d'automatiser partiellement l'acquisition de connaissance sur le trafic. Elles sont cependant tributaires de l'existence de données nécessaires pour l'apprentissage.

En dehors de la simple problématique de la détection des anomalies du trafic, ces systèmes ne fournissent que peu d'informations sur les anomalies détectées. L'opérateur en charge de la surveillance du réseau ne possède pas d'information quant à la hiérarchie en termes de dangerosité des anomalies détectées. Il doit aussi procéder à une fastidieuse tâche d'analyse et de caractérisation des anomalies détectées. Ces deux tâches de traitement d'anomalies ne sont pas traitées par les systèmes de détection d'anomalies du trafic existants.

A.1.3 Détection autonome d'anomalies du trafic

L'hypothèse originale utilisée dans les systèmes de détection d'anomalies du trafic est la suivante :

Hypothesis A.1 *Le trafic anormal est différent statistiquement du trafic normal. [21, 22]*

Comme nous l'avons énoncé dans la section A.1.2, les systèmes de détection d'anomalies existants utilise deux approches : la détection par signatures et la détection d'anomalie. Ces deux approches utilisent l'hypothèse A.1 pour construire des modèles du trafic anormal (détection basée sur les signatures) ou du trafic normal (détection d'anomalie). Cependant, et comme nous l'avons vu dans la section A.1.1, le trafic réseau est extrêmement changeant. Les systèmes de détection d'anomalie du trafic sont vulnérables à ces changements.

Nous pensons donc que les systèmes de détection d'anomalie du trafic doivent être efficaces malgré ces changements. Ces systèmes ne doivent pas utiliser de connaissances détaillées sur le trafic ou la structure des anomalies. Ils doivent plutôt utiliser une seule hypothèse pour séparer le trafic normal du trafic anormal :

Hypothesis A.2 *La majorité du trafic est normal.*
Ce qui est équivalent à : seulement $X\%$ des représentations du trafic sont anormales avec $X < 50\%$. [23]

Les systèmes de détection d'anomalies efficaces doivent donc utiliser les hypothèses A.1 et A.2 afin de trouver les anomalies du trafic de façon autonome. Les techniques d'apprentissage non-supervisé permettent d'extraire des groupes d'instances similaires et de séparer les groupes d'instances différentes entre elles. Ces techniques permettent donc aux systèmes de détections d'anomalies du trafic de séparer les trafics anormaux et normaux sous réserve que l'hypothèse A.1 soit vérifiée. Une fois que les trafic normaux et anormaux sont séparés, l'hypothèse A.2 nous fournit une critère pour identifier le trafic normal et par extension le trafic anormal.

Nous pensons que ce principe de détection d'anomalies du trafic est la voie à suivre pour construire des systèmes de détection d'anomalie du trafic ne reposant pas sur des connaissance acquises et robuste vis-à-vis de l'évolution, la disparité et la mutation du trafic.

A.1.4 Résumé

Le trafic change au cours du temps selon les points de mesure utilisés. Nous pensons que les systèmes de détection d'anomalies existants qui s'appuient sur des modèles pour le trafic normal ou anormal sont inefficaces vis-à-vis de ces phénomènes. Les coûts de mise au point et de mise à jour de ces modèles compliquent leur utilisation dans le monde réel. Nous pensons donc que les systèmes de détection des anomalies du trafic doivent être capables de s'adapter de manière autonome au changement du trafic via l'apprentissage non-supervisé.

A.2 Contributions

Notre contribution comprend deux parties. La première partie est l'application des techniques de partitionnement à la détection d'anomalies. La deuxième partie est constituée par un ensemble de techniques de traitement des anomalies.

A.2.1 Détection non-supervisée des anomalies du trafic

Cette partie propose tout d'abord plusieurs techniques pour traiter le trafic réseau : l'agréger en flux et construire des attributs. Nous expliquons ensuite comment les différentes parties du trafic sont représentées au sein de l'espace constitué des attributs précédemment construits. Nous proposons ensuite un technique de partitionnement qui nous permet d'extraire les flux anormaux du reste du trafic.

A.2.2 Traitement des anomalies

Cette partie propose différentes techniques qui traitent le résultat de l'étape précédente. La première de ces techniques corrèle les anomalies extraites lors de la détection non-supervisée. La seconde technique utilise les résultats de l'étape de corrélation et deux caractéristiques de chaque anomalie pour les classer. La troisième et dernière technique caractérise les anomalies extraites via la construction de signature.

A.3 Contexte au sein du projet ECODE

Cette thèse a été réalisée dans le cadre du projet ECODE. Ce projet est financé par la Commission Européenne via le contrat FP7-ICT-2007-2/223936. Le but de ce projet est d'introduire des capacités cognitives dans les équipements réseaux.

A.4 Plan du manuscrit

Ce manuscrit est composé de cinq chapitres :

- Le chapitre B présente un état de l'art dans les domaines de l'apprentissage non-supervisé et de la détection d'anomalies du trafic.

- Le chapitre C aborde le principe de notre technique de détection d'anomalies du trafic basée sur l'apprentissage non-supervisé.
- Le chapitre D présente plusieurs techniques qui traitent les résultats de notre détection dans le but de faciliter le travail de l'opérateur.
- Le chapitre E expose l'évaluation de notre algorithme sur du trafic réel.
- Le chapitre F conclut ce manuscrit en résumant notre contribution et en proposant plusieurs pistes quant à l'amélioration de notre algorithme.

Chapitre B

État de l'art

Ce chapitre introduit différents travaux réalisés dans les domaines de l'apprentissage non-supervisé et de la détection d'anomalie. La première section aborde les techniques d'apprentissage non-supervisée. La seconde section présente différents algorithmes dédiés à la détection des anomalies du trafic.

B.1 Apprentissage non-supervisé

Les techniques d'apprentissage non-supervisé visent à extraire une structure cachée à l'intérieur de données inconnues. Les méthodes de séparation aveugle de source, les réseaux de neurones et le partitionnement sont des techniques permettant l'apprentissage non-supervisé.

Séparation aveugle de source Les méthodes de séparation aveugle de source visent à extraire un ensemble de signaux à partir d'un ensemble de signaux mixés. Plusieurs techniques traitent ce problème : Analyse en Composantes Principales (ACP), Analyse en Composantes Indépendantes (ACI).

Réseaux de neurones Les réseaux de neurones constituent le second type de méthodes d'apprentissage non-supervisé présentées dans ce manuscrit. Une liste non-exhaustive des réseaux de neurones utilisés pour l'apprentissage non-supervisé est : la Théorie de la Résonance Adaptative (TRA) et les Cartes Auto-Adaptatives (CAA).

Partitionnement Les techniques de partitionnement visent à grouper des éléments similaires entre eux selon une mesure de similarité. Les techniques de partitionnement se divisent en deux grandes catégories principales.

La première catégorie contient les algorithmes de partitionnement hiérarchique. Ces algorithmes visent à créer une hiérarchie entre les groupes d'instances (ou *cluster*). Le partitionnement final est obtenu en fixant un niveau dans la hiérarchie et en gardant les groupements existants au niveau considéré.

La seconde catégorie regroupes les algorithmes de partitionnement dit ?de partition? par opposition au partitionnement hiérarchique. Ces algorithmes visent à construire une partition globale en une seule passe. Parmi les algortihmes existants de cette famille, nous pouvons citer : k-means [28], DBSCAN [29], etc. Plusieurs travaux récents ont tenté d'appliquer le partitionnement à des données de grandes dimensions. Nous pouvons ici citer MAFIA [39] ou MineClus [46, 47]. Dans [30], Kriegel et al. réalisent un état de l'art plus détaillé de ces techniques. Dans [31], [77] et [74], les auteurs évaluent plusieurs algorithmes de cette catégorie.

B.2 Détection des anomalies du trafic réseau

La problématique de la détection d'anomalies a donné lieu a un grand nombre de publications au cours de la dernière décennie. La plupart des approches existantes analysent les variations statistiques de métriques simples (e.g. : nombre de paquets, d'octets, ou de flux) et/ou d'autres attributs du trafic (e.g. : distribution des adresses IP et des ports) et ce, sur des données issues d'un seul lien ou de l'ensemble d'un réseau. Parmi les outils utilisés, nous pouvons citer les techniques d'analyse du signal (e.g. : Auto-Regressive Integrated Moving Average, ondelettes) sur des données issues d'un seul lien [95, 17], la PCA [104, 18] et les filtres de Kalman [91] pour la détection sur l'ensemble d'un réseau et les *sketches* appliqués aux flux IP [89, 92].

Notre approche peut être classifiée dans le champ de la détection non-supervisée. La plupart des travaux existants utilisant des techniques non-supervisées ont traité le domaine de la détection d'intrusions à travers la base de données KDD'99. La plupart des méthodes de détection non-supervisées utilisent le *clustering* et la détection d'outliers (points isolés) [23, 124, 125]. Dans [23], Portnoy et al. utilisent du *single-linkage hierarchical clustering* pour partitionner les données KDD'99 en utilisant une distance euclidienne. Dans [124], Eskin et al. présentent des résultats améliorés sur les mêmes données obtenus avec trois algorithmes de partitionnement : un *fixed-width clustering*, une version optimisée de k-NN, et un one class *support vector machine* (machine à vecteurs de support). Dans [125], Leung et al. utilisent un algorithme de *density-grid-based clustering* (partionnement basé sur la densité et sur les grilles) pour améliorer l'extensibilité du *clustering* et obtient des résultats équivalents.

B.3 Résumé

Les techniques d'apprentissage supervisé existent depuis longtemps. Elles visent a extraire de l'information sur des données sans avoir aucune connaissance préalable sur ces même données.

Nous avons aussi présenté les techniques existantes en matière de détection des anomalies du trafic. Nous présentons plusieurs algorithmes existants qui utilisent l'apprentissage non-supervisé dans le domaine de la détection d'intrusion.

Le chapitre suivant présente notre propre application de l'apprentissage non-supervisé à la problématique de la détection des anomalies du trafic.

Chapitre C

Détection non-supervisée des anomalies

Ce chapitre présente notre algorithme de détection non-supervisé des anomalies du trafic. Comme cela est annoncé dans les sections A.1.3 et A.2, notre système vise à extraire les flux anormaux en utilisant peu de connaissance sur la structure du trafic. Ce but est atteint via l'utilisation d'algorithmes de partitionnement et des hypothèses énoncés dans la section A.1.3.

Notre algorithme est divisé en trois étapes. La première étape est une étape de traitement des données qui est directement appliquée au trafic réseau. Cette étape applique tout d'abord un algorithme de détection de changement à des séries basées sur le trafic. Les fenêtres temporelles jugées anormales sont ensuite traitées afin de construire des flux et attributs sur ces mêmes flux. Cette partie de notre algorithme est située en haut de la figure C.1 et à l'intérieur du cadre en pointillé d'indice 1. La détection non-supervisée constitue la seconde étape de notre algorithme. Cette étape est visible sur la partie basse de la figure C.1 et est entourée par le cadre en pointillé d'indice 2. Les concepts présentés dans ce chapitre ont été publiés dans [127, 128, 129, 130, 131] et exposés dans les rapports ECODE D3.2 [132] et D3.3 [133]. La troisième étape de notre algorithme traite les flux anormaux extraits par la détection non-supervisée. Cette étape est présentée dans le chapitre suivant. Elle est située à l'intérieur du cadre d'indice 3 dans la figure C.1.

C.1 Détection de changement, agrégation en flux multi-résolution et construction d'attributs

La première phase de l'analyse est la détection de changement. Pour ce faire, les séries temporelles Z^i sont construites pour trois métriques : le nombre de paquets, le nombre d'octets et le nombre de paquets SYN.

Un algorithme de détection d'anomalies $\mathcal{F}(.)$ basé sur l'analyse de séries temporelles est utilisé sur les Z^i afin d'identifier une fenêtre anormale. Z^i représente la série temporelle pour la métrique i. Z^i_t représente la valeur de la métrique i pour la fenêtre t. Une fenêtre t_j est déclarée anormale si $\mathcal{F}(Z^i_t)$ déclenche une alarme. $\mathcal{F}(.)$ utilise ici des deltoïdes tels que Cormode et al. les ont définis dans [88].

Le trafic est ensuite agrégé suivants 9 niveaux l_i : *adresse IP source* (l_1 : IPsrc), *adresse*

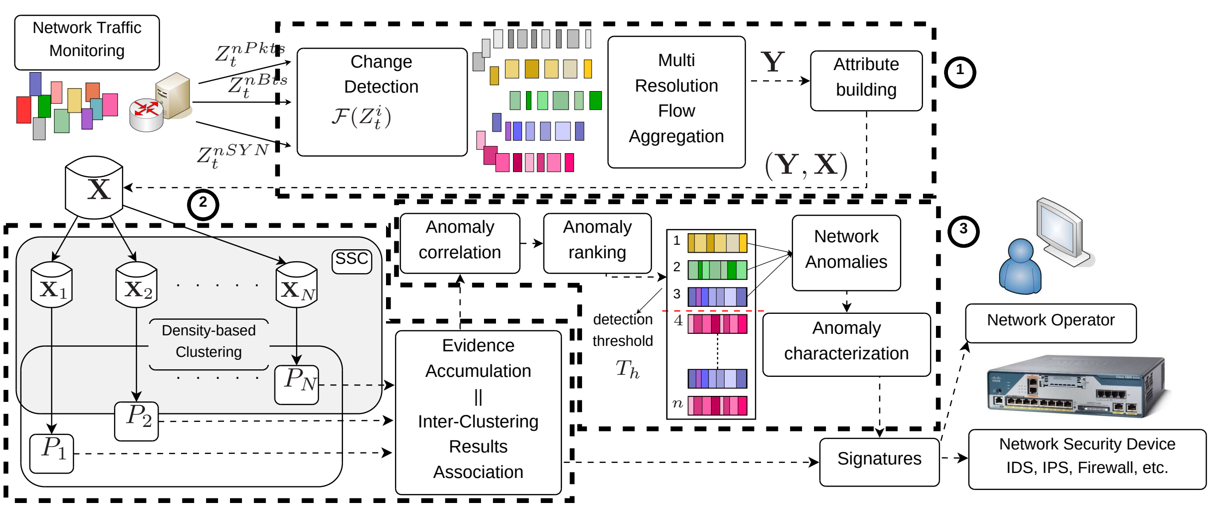


FIGURE C.1 – Schéma global de notre approche.

IP destination (l_2 : IPdst), *préfixe réseau source* ($l_{3,4,5}$: IPsrc/24, /16, /8), *préfixe réseau destination* ($l_{6,7,8}$: IPdst/24, /16, /8), et *trafic par fenêtre* (l_9 : tpTS).

Des attributs sont ensuite construits pour chacun des flux afin de les caractériser. Le tableau C.1 liste ces attributs.

Attribute	**Description**
nDests	# adresses IP destination différentes
nSrcs	# adresses IP source différentes
nPkts/nDiffDestPort	# paquets divisé par # ports destination différents
nDiffSrcAddr/nDiffDestAddr	# adresses IP source différentes divisé par # adresses IP destination différentes
nICMP/nPkts	# paquets icmp divisé par # paquets
nEchoReqReply/nPkts	# paquets ICMP echo-request ou reply divisé par # paquets
nSYN/nPkts	# paquets syn divisé par # paquets
nRST/nPkts	# paquets rst divisé par # paquets
bgstDestPort/tNbOccuDestPort	# occurrence du port destination avec la plus grand occurrence divisé par # total de port destination

TABLE C.1 – Attributs sur les flux agrégés dérivés du trafic.

C.2 Comment détecter des anomalies du trafic via l'apprentissage non-supervisé ?

La deuxième étape de l'algorithme consiste à détecter et caractériser de manière non-supervisée les flux anormaux. L'algorithme utilise pour cela l'ensemble $\mathbf{Y} = \{\mathbf{y}_1, .., \mathbf{y}_F\}$ des F flux capturés dans l' intervalle anormal, agrégés suivant l'un des niveaux d'agrégation l_i utilisés dans la première phase. Chaque flux $\mathbf{y}_f \in \mathbf{Y}$ est décrit par un ensemble de A attributs de trafic sur lesquels l'analyse est faite. $\mathbf{x}_f = (x_f(1), .., x_f(A)) \in \mathbb{R}^A$ est le vecteur des attributs décrivant le flux de trafic $\mathbf{y}_f$, et $\mathbf{X} = \{\mathbf{x}_1; ..; \mathbf{x}_F\}$ la matrice des attributs appelée aussi espace d'attributs.

C.2.1 Représentations du trafic réseau

Cette section abord les représentations du trafic dans la matrice d'attributs. Les deux sous-parties détaillent respectivement les représentations des trafics normaux et anormaux.

C.2.1.1 Représentations du trafic normal

La section C.1 spécifie que la représentation atomique du trafic est un flux agrégé. C'est dans ce contexte que l'hypothèse A.2 est raffinée :

Hypothesis C.1 *La majorité de flux agrégés contient du trafic normal. Ce qui est équivalent à : $Y\%$ des flux agrégés sont normaux avec $Y > 50\%$.*

L'hypothèse C.1 spécifie que la majorité des flux agrégés sont normaux. L'hypothèse A.1 dit que les anomalies sont statiquement différentes du trafic normal. Si l'on considère la matrice d'attributs en entier, le trafic normal peut être vu comme un ou plusieurs *cluster(s)* qui contiennent au moins la moitié des flux agrégés. Cette hypothèse est raffinée dans la section C.3.

C.2.1.2 Représentations du trafic anormal

Pour cela, nous faisons l'hypothèse que l'anomalie est contenue dans un petit nombre de flux. Ainsi, une anomalie peut apparaître comme un outlier (i.e. un flux isolé) ou un *cluster* de petite taille (quelques flux similaires), et ce, selon le type d'agrégation et le masque réseau utilisés. Le tableau C.2 détaille les caractéristiques de différentes anomalies : nature distribuée, type d'agrégation et masque de réseau utilisés et signature construite. Une anomalie de type SYN DDos (SYN Distributed Denial of Service ou attaque de déni de service distribuée) qui vise une seule cible depuis un grand nombre d'adresses IP situées dans plusieurs domaines /24 formera un *cluster* si le trafic est agrégé avec une IP source. En effet, chacun des domaines /24 représentera un flux avec des valeurs d'attributs très éloignées de celles de la majorité du trafic : un nombre important de paquets envoyés, une seule destination et une très forte proportion de paquet SYN. C'est l'ensemble de ces flux qui formera un *cluster*. Si le trafic est agrégé via l'adresse IP destination, quelque soit le masque, l'unique adresse destination (i.e. la cible) formera un outlier qui se caractérisera notamment par un grand nombre de sources et une forte proportion de paquets SYN.

C.2.2 Présentation générale

Nous avons développé une approche de *clustering* de type « division et conquête » qui se base sur les notions de *clustering ensemble* [137] et de *clustering combination*. Cette approche combine le *density-based clustering* [29], le *sub-space clustering* (SSC)[31] et l'*evidence accumulation* (EA) [139]. Nous présentons ici l'idée générale de l'approche.

Au lieu de directement partitionner l'espace des attributs $\mathbf{X}$ en entier en utilisant une distance classique comme la distance euclidienne, l'algorithme partitionne N sous-espaces différents $\mathbf{X}_n \subset \mathbf{X}$, ce qui permet ainsi d'obtenir N résultats de partitionnement P_n des flux de $\mathbf{Y}$. Chaque sous-espace $\mathbf{X}_n$ est construit à partir de $R < A$ attributs.

TABLE C.2 – Attributs utilisés pour la détection de DoS, DDoS, balayage de ports et de réseaux (port scan and network scan), et propagation de vers. (NB : les anomalies de nature distribuée impliquent ici plusieurs @IP/24, elles-mêmes contenues dans une seule @IP/16)

Exemple d'anomalie	Nature distribuée	Type d'agrégat. /masque réseau	Résultat de clustering	Impact sur les attributs de trafic
DoS (ICMP$\vee$SYN)	1-vers-1	IPsrc/$*$	Outlier	nSrcs = nDsts = 1, nPkts/sec $> \lambda_1$, avgPktsSize $< \lambda_2$, (nICMP/nPkts $> \lambda_3 \vee$ nSYN/nPkts $> \lambda_4$).
		IPdst/$*$	Outlier	
DDoS (ICMP $\vee$ SYN) depuis plus. @IP/24	N-vers-1	IPsrc/24 (l_3)	Cluster	nDsts = 1, nSrcs $> \alpha_1$, nPkts/sec $> \alpha_2$, avgPktsSize $< \alpha_3$, (nICMP/nPkts $> \alpha_4 \vee$ nSYN/nPkts $> \alpha_5$).
		IPsrc/16 (l_4)	Outlier	
		IPdst/$*$	Outlier	
Port scan	1-vers-1	IPsrc/$*$	Outlier	nSrcs = nDsts = 1, nDstPorts $> \beta_1$, avgPktsSize $< \beta_2$, nSYN/nPkts $> \beta_3$.
		IPdst/$*$	Outlier	
Network scan vers plusieurs @IP/24	1-vers-N	IPsrc/$*$	Outlier	nSrcs = 1, nDsts $> \delta_1$, nDstPorts $> \delta_2$, avgPktsSize $< \delta_3$, nSYN/nPkts $> \delta_4$.
		IPdst/24 (l_6)	Cluster	
		IPdst/16 (l_7)	Outlier	
Propagation de ver sur plusieurs @IP/24	1-vers-N	IPsrc/$*$	Outlier	nSrcs = 1, nDsts $> \eta_1$, nDstPorts $< \eta_2$, avgPktsSize $< \eta_3$, nSYN/nPkts $> \eta_4$.
		IPdst/24 (l_6)	Cluster	
		IPdst/16 (l_7)	Outlier	

Cela permet d'analyser la structure de $\mathbf{X}$ depuis R-parmi-A différents points de vue tout en utilisant une bien meilleure résolution. Pour déterminer le nombre R de dimensions utilisées par sous-espace, nous utilisons la propriété monotone des ensembles de résultats de partitionnement appelée *downward closure property* : "si un ensemble de points forme un *cluster* dans un espace de dimension d, alors, ce même ensemble de points forme aussi un *cluster* dans chacune des projections de dimension $d-1$ de ce même espace". Cela implique que, s'il existe des *clusters* dans $\mathbf{X}$, alors, ils existeront dans des sous-espaces de dimensions inférieures. L'utilisation de petites valeurs de e (ou R dans notre cas) présente alors plusieurs avantages : premièrement, le partitionnement est plus rapide dans des espaces de faibles dimensions. Deuxièmement, les algorithmes de partitionnement basés sur la densité sont plus efficaces dans des espaces de dimensions réduites [38]. Enfin, les résultats de partitionnement obtenus sont plus faciles à visualiser. Cet ensemble de contraintes nous a amené à choisir un nombre R de dimensions utilisées égal à 2, et ainsi $N(m) = C_R^A = A(A-1)/2$.

Accumulation de Preuves L'information obtenue entre les différents résultats de partitionnement construits pour $\mathbf{X}_n$ est ensuite combinée afin de produire une nouvelle mesure de similarité entre les flux de $\mathbf{Y}$. Ces valeurs sont stockées dans une matrice S de taille $F \times F$ qui sert à détecter les petits *clusters* et un vecteur D qui est utilisé pour classer les outliers. L'élément $S(o,p)$ représente le degré de similitude entre les flux o et p. Cette valeur reflète le nombre de fois où les flux o et p considérés sont dans le même *cluster*. Elle tient aussi compte de la taille du *cluster* en question afin de privilégier les petits *clusters*. L'élément $D(d)$ reflète quant à lui l'anormalité d'un outlier. Cette anormalité tiens compte du nombre de fois où le flux a été classé outlier et de la séparation entre cet outlier et le reste du trafic. Cela permet de séparer les *outliers* de flux et les *clusters* de flux simultanément identifiés dans différents sous-espaces du reste du trafic. Ces nouvelles mesures de similarité permettent donc d'extraire les flux anormaux de l'ensemble du trafic. En d'autres termes, si nous pouvons extraire des flux qui se démarquent de la majorité du trafic, alors, ces flux forment une anomalie, sinon, cela signifie que l'alarme sur la fenêtre temporelle considérée $Z_{t_j}^{l_i}$ était une fausse alarme.

Association Inter-Résultats de Partitionnement Cependant, en raisonnant sur les similarités entre instances (ici flux), l'accumulation de preuve introduit plusieurs sources d'erreurs. Si l'on considère deux ensembles d'instances P_i et P_j, si la cardinalité de ces deux ensembles est proches et si ils sont présents dans un nombre similaires de sous-espace, alors, l'accumulation de preuves produira des valeurs de similarités très proches pour chacun des flux appartenant aux deux ensembles. Ces flux seront alors considérés comme appartenant au même *cluster*. Ce cas de figure peut être dangereux dans la mesure ou deux anomalies pourraient être groupées ensemble et donc être mal identifiées et par la suite, mal caractérisées. Un autre type d'erreur peut survenir si l'utilisation d'une algorithme de partitionnement sur les valeurs de S est faite avec de mauvais paramètres. De plus, l'utilisation de seuils sur S et/ou D peut sensiblement dégrader les performances de notre système si de mauvaises valeurs sont utilisées.

Afin d'éviter ces problèmes, nous proposons une nouvelle approche pour combiner les résultats de partitionnement obtenu dans chaque sous-espace : Association Inter-Résultats de Partitionnement. L'idée est ici d'aborder le problème en termes de similarité de *cluster* et d'*outlier* au lieu d'utiliser la similarité entre les instances (ici les flux). Nous déplaçons ainsi la mesure de similarité des instances vers les résultats de partitionnement. Le problème peut donc être sub-divisé en deux sous-problèmes : corréler les *clusters* via l'Association INter-Cluster (AINC) et corréler les *outliers* via l'Association INter-Outlier (AINO).

Dans chacun de ces cas, nous exprimons la similarité entre résultats de partitionnement via un graphe. Chaque sommet est un *cluster* ou un *outlier* issu de n'importe lequel des sous-espace $\mathbf{U}_n$ et l'existence d'un arc entre deux sommets veut dire que les résultats de partitionnement représentés par ces sommet sont similaires. Une fois construits, nous devons analyser ces graphes afin de trouver des ensembles de résultats de partitionnement similaires entre eux. En termes de sommets, ces ensembles de résultats de partitionnement similaires entre eux se traduisent par des ensembles de sommets pour lesquels chaque sommet est lié à tous les autres sommets de l'ensemble considéré. La théorie des graphes assigne le terme de clique à ce type d'ensemble de sommets. Le problème de la recherche de la clique maximale est NP-complet. La plupart des solutions existantes utilisent une recherche exhaustive de sous-graphes à l'intérieur du graphe. Cette solution est malheureusement trop lente pour nos besoins. Nous utilisons donc un algorithme glouton pour identifier les cliques au sein des graphes de similarité. Chaque ensemble de flux est appelé un segment de trafic. Il est construit à partir de l'intersection des flux contenus dans chaque *cluster* ou *outlier* de la clique considérée.

C.3 Redéfinition de la représentation du trafic normal

A la lumière de la présentation de notre technique de partitionnement, dans les sections C.2.2, et pour faciliter l'automatisation du processus de détection d'anomalie, nous redéfinissons la représentation du trafic normal de l'hypothèse A.2 :

Hypothesis C.2 *La majorité des flux agrégés sont normaux. Ces flux forment un seul* cluster *qui contient au moins Y% des flux agrégés.*

La *downward closure property* [38], nous permet d'étendre la définition de la représentation du trafic normal présenté dans la section C.2.1. Étant donné que le trafic normal est représenté par un seul *cluster* qui contient au moins la moitié des F flux, la propriété énoncée plus haut garantit que ce cluster sera aussi présent dans chacun des $\mathbf{X}_n$ sous-espaces. Nous vérifions donc ce constat en deux points de notre algorithme et faisons donc deux nouvelles hypothèses.

Hypothesis C.3 *Dans chaque sous-espace* $\mathbf{X}_n$*, le plus gros* cluster *(en termes de nombre de flux agrégés)* doit *contenir au moins Y% des flux agrégés.*

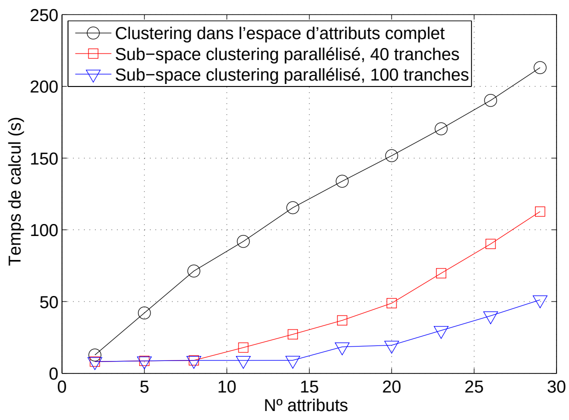
Hypothesis C.4 *Dans chaque matrice d'attributs, le plus gros segment de trafic (en termes de nombre de flux agrégés)* doit *contenir au moins Y% des flux agrégés.*

Si une de ces deux hypothèses n'est pas vérifiée, la détection s'arrête et UNADA recommence sur la fenêtre temporelle suivante.

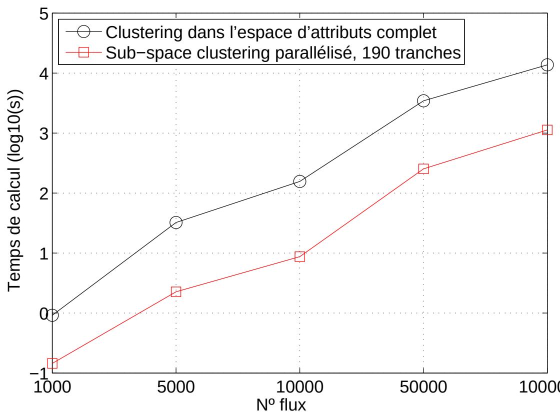
C.4 Temps de calcul et parallélisation

Nous abordons maintenant le temps de calcul (TC) de l'algorithme. Notre algorithme réalise plusieurs partitionnements sur $N(m)$ sous-espaces $\mathbf{X}_n \subset \mathbf{X}$ de petite dimension. Ces multiples calculs posent le problème de l'extensibilité pour une détection temps-réel sur des cœurs de réseaux à très hauts débits. Deux caractéristiques de notre algorithme sont utilisées pour améliorer ces problèmes d'extensibilité vis-à-vis du nombre d'attributs m et du nombre de flux agrégés n à analyser. D'une part, le *clustering* est réalisé dans des sous-espaces de très petites dimensions, $\mathbf{X}_n \in \mathbb{R}^R$ avec ici $R = 2$ (cf. section C.2.2), ce qui est plus rapide qu'un *clustering* réalisé dans un espace de grande dimension [138]. D'autre part, chaque sous-espace peut être partitionné indépendamment des autres sous-espaces, ce qui rend l'algorithme de *subspace* clustering particulièrement adapté à la parallélisation. Cette parallélisation peut être réalisée de différentes façons : via une machine multi-cœurs et/ou multi-processeurs, via une ou des carte(s) réseaux avec processeurs de traitement intégré, via un ou des GPU(s) (Graphic Processor Unit), via plusieurs machines ou via une combinaison de n'importe lesquelles de ces techniques. Nous utiliserons désormais le terme tranche pour nommer une entité de calcul.

La figure C.2 représente le temps de calcul de notre algorithme, en fonction du nombre d'attributs m utilisés pour décrire les flux (a) et en fonction du nombre n de flux à analyser (b). La figure C.2.(a) compare le temps de calcul obtenu lorsque l'on partitionne l'espace d'attributs $\mathbf{X}$, noté TC($\mathbf{X}$) avec le temps de calcul nécessaire au *sub-space clustering*, en faisant varier A de 2 à 29 attributs. Un grand nombre de flux N, ici 10^4, est analysé en utilisant deux nombres de tranches $T = 40$ et $T = 100$. Pour estimer le temps de calcul de notre algorithme de *sub-space clustering* avec des valeurs données de m et T, nous procédons de la façon suivante : premièrement, nous partitionnons chacun des



(a) Temps vs. n attributs.



(b) Temps vs. n flux.

FIGURE C.2 – Temps de calcul en fonction du nombre d'attributs et de flux à analyser. Le nombre de flux dans (a) est $n = 10000$. Le nombre d'attributs et de tranches pour (b) est $m = 20$ et $M = 190$.

$N = m(m-1)/2$ sous-espaces $\mathbf{X}_n$ et le plus mauvais résultat en terme de temps de calcul pour un seul sous-espace $\mathrm{TC}(\mathbf{X}_{\mathrm{SSCwc}}) = \max_n \mathrm{TC}(\mathbf{X}_n)$ est conservé en tant que référence. Si $N \leqslant T$, le nombre de tranches couvre la totalité des partitionnements à réaliser, l'algorithme va donc s'exécuter de manière complètement parallèle. Le temps de calcul total correspond au plus mauvais résultat $\mathrm{TC}(\mathbf{X}_{\mathrm{SSCwc}})$. Par contre, si $N > T$, certaines tranches doivent partitionner plusieurs sous-espaces, le temps de calcul devient égal à $(N\%T+1)$ fois le plus mauvais cas $\mathrm{TC}(\mathbf{X}_{\mathrm{SSCwc}})$, où $\%$ représente la division entière. La première observation intéressante sur la figure C.2.(a) concerne l'augmentation de $\mathrm{TC}(\mathbf{X})$ quand m augmente ; TC vaut 8 secondes pour $m = 2$ et plus de 200 secondes for $m = 29$. La diminution du TC pour des petites valeurs de m permet de compenser en partie le grand nombre de partitionnements réalisés. La seconde observation concerne le parallélisme : si l'algorithme est déployé dans une architecture parallèle, il peut être utilisé pour analyser de grandes quantités de trafic en utilisant un grand nombre d'attributs en temps-réel. Par exemple, l'analyse de 20 attributs sur une architecture avec 100 tranches permet de traiter 10000 flux en moins de 20 secondes.

La figure C.2.(b) compare $\mathrm{TC}(\mathbf{X})$ et $\mathrm{TC}(\mathbf{X}_{\mathrm{SSCwc}})$ en fonction du nombre de flux n à analyser en utilisant $m = 20$ attributs et $M = N = 190$ tranches (i.e. notre algorithme est complètement parallélisé). La différence en termes de temps de calcul entre le *clustering* dans l'espace entier et le *clustering* dans les sous-espaces est toujours présente : le gain est de plus d'un ordre de magnitude, indépendamment du nombre de flux à analyser. En ce qui concerne la quantité de trafic analysable dans une configuration complètement parallèle, l'algorithme est capable de traiter plus de 50000 flux avec un temps de calcul relativement limité : 4 minutes environ. Ces évaluations ont été réalisées avec un nombre de flux agrégés égal à environ 2500 dans une fenêtre temporelle de durée $\Delta T = 20s$, ce qui représente un temps de calcul de $\mathrm{TC}(\mathbf{X}_{\mathrm{SSCwc}}) \approx 0.4$ secondes. Si l'on considère les $m = 9$ attributs utilisés précédemment ($N = 36$), le temps de calcul total sans parallélisation est de $N\times$ $\mathrm{TC}(\mathbf{X}_{\mathrm{SSCwc}}) \approx 14.4$ secondes.

C.5 Résumé

Dans ce chapitre, nous avons présenté notre algorithme de détection non-supervisé. UNADA utilise deux hypothèses : le trafic anormal est différent du trafic normal et le trafic normal est plus important que le trafic anormal. Nous utilisons du partitionnement afin d'isoler les composantes normales et anormales du trafic réseau. L'évaluation du temps d'exécution de notre algorithme prouve que son extensibilité est bonne. Notre algorithme peut donc être efficacement parallélisé.

Une fois que les anomalies ont été extraites, une étape de traitement est nécessaire afin de réduire la charge de travail de l'opérateur. Cette étape est abordée dans le chapitre suivant.

Chapitre D

Traitement des anomalies

Le partionnement non-supervisé utilisé dans le chapitre C nous permet d'extraire les flux anormaux parmi les F flux agrégés construits selon un niveau d'agrégation l_i. Cependant cette étape peut potentiellement extraire un grand nombre de flux. Le danger est ici de submerger l'opérateur avec un grand nombre d'alarmes. Une étape de traitement des anomalies extraites se justifie donc pleinement dans le but d'aider l'opérateur à traiter en priorité les anomalies les plus dangereuses. Une aide doit aussi être apportée à l'opérateur dans le domaine de l'analyse des anomalies.

Nous proposons donc une technique de traitement des anomalies en trois étapes afin de réduire le travail de l'opérateur. Tout d'abord, nous réduisons le nombre total d'anomalies en corrélant les anomalies extraites de flux agrégés suivant plusieurs niveaux d'agrégation l_i. Ensuite, nous utilisons une technique qui construit une hiérarchie entre les anomalies en termes de dangerosité et permet donc à l'opérateur de prioritiser son travail. Enfin, nous appliquons plusieurs heuristiques pour construire des signatures d'anomalies. Chacune de ces étapes sont définies sur la figure D.1.

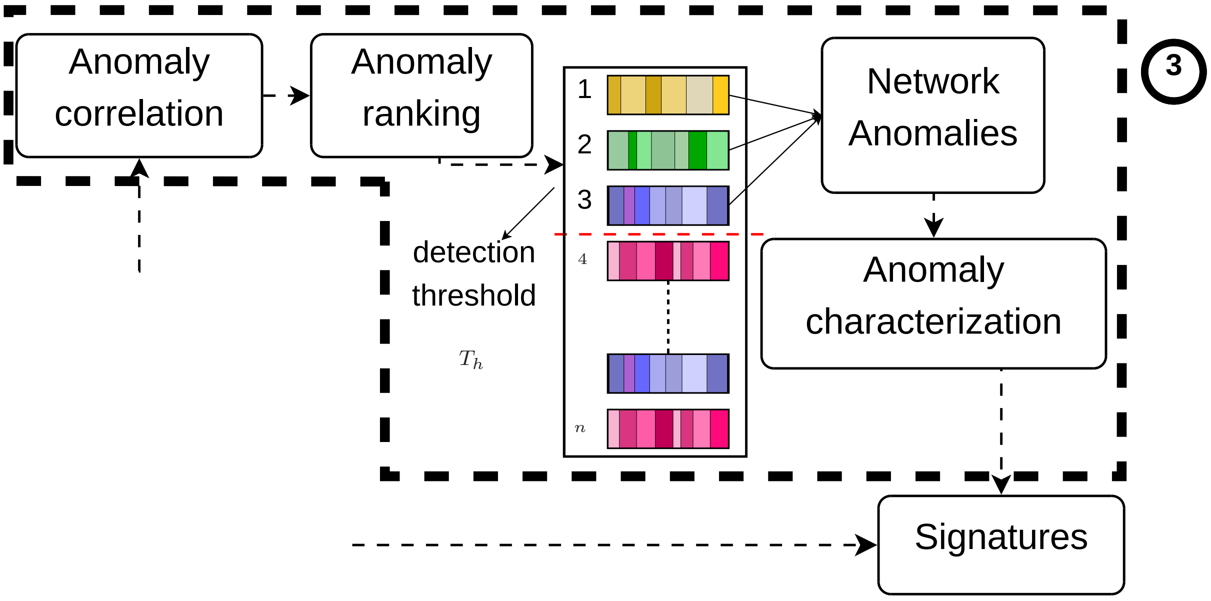


FIGURE D.1 – Schéma de l'étape de traitement des anomalies.

D.1 Correlation d'anomalies

Dans le chapitre précédent nous avons vu que notre algorithme non-supervisé est capable d'extraire les anomalies des flux agrégés selon un certain niveau d'agrégation l_i. La possibilité d'utiliser notre algorithme sur plusieurs niveaux d'agrégations nous amènent à penser qu'utilisé ainsi, il générera un grand nombre d'alarmes. Une interrogation émerge donc : pourquoi ne pas corréler les anomalies extraites d'ensembles de flux construits sur différents niveaux d'agrégation l_i afin de de réduire le nombre d'alarmes tout en améliorant la fiabilité de nos résultats ? Nous suivons donc ce raisonnement en obtenant plusieurs résultats de partitionnement, chacun étant construit à partir de plusieurs niveaux d'agrégations, et en élaborant une stratégie pour corréler les anomalies extraites de ces résultats de partitionnement. Cette méthode a été publiée dans [131].

Pour corréler les anomalies trouvées dans différents niveaux d'agrégations, nous définissons deux caractéristiques uniques d'une anomalie : son ensemble d'adresses source et son ensemble d'adresses destination. Deux anomalies sont considérées comme similaires si ces deux ensembles d'adresses sont similaires.

Dans ce travail, nous nous limitons à la corrélation d'anomalies détectées selon deux types d'agrégations : adresse IP source et adresse IP destination. Nous n'utilisons donc pas d'anomalies trouvées dans les même types de niveaux d'agrégation (l_1 et l_2 par exemple) car ces anomalies pourraient être incluses entre elles. De plus, nous nous limitons à un masque réseau de /24 pour les niveaux d'agrégations. Ces deux choix ont été faits afin de réduire la quantité de calcul nécessaire. Les niveaux d'agrégations utilisés sont donc l_3 et l_6 (cf. niveaux d'agrégations dans la section C.1).

L'ensemble des anomalies corrélées est obtenu à partir des couples d'anomalies similaires.

D.2 Hiérarchisation des anomalies

L'opérateur a besoin de prioritiser ses différentes tâches afin d'augmenter son efficacité. Dans le domaine de la détection d'anomalie du trafic, plus une anomalie est dangereuse, plus son analyse est critique. La prioritisation des tâches dans ce contexte requiert donc une hiérarchisation des anomalies en fonction de leur dangerosité.

A la suite de l'étape de corrélation d'anomalies, plusieurs anomalies ont été corrélées. Ces anomalies nous fournissent un indice prima facie d'une hiérarchie entre les anomalies en termes de dangerosité. En effet, si une anomalie apparaît en tant que telle dans plusieurs niveaux d'agrégations, cela veut dire que ses flux sont différents du trafic normal dans chacun de ces niveaux d'agrégations, et donc, que l'anomalie en question est potentiellement dangereuse dans chacun d'entre eux. Les anomalies corrélées peuvent donc être considérées comme étant plus dangereuses.

Nous introduisons donc un indice de dangerosité basé sur le nombre de niveaux d'agrégation dans lesquels une anomalie est détectée. Cet indice tiens également compte de la proportion de trafic contenu dans les flux anormaux de l'anomalie considérée en termes de nombres de paquets et de nombre d'octets.

Le seuil T_h dans la figure D.1 est un seuil sur la valeur de l'indice de dangerosité de l'anomalie considérée. Cette méthode a été partiellement publiée dans [131].

D.3 Caractérisation automatique des anomalies

A la fin de l'étape précédente, l'algorithme non supervisé a identifié un ou plusieurs flux similaires dans $\mathbf{Y}$ et éloignés de la majorité des flux du trafic. La tâche suivante consiste à produire automatiquement un ensemble de règles de filtrage pour isoler et caractériser ce(s) flux. Ces règles de filtrage permettent deux choses : obtenir une image précise de la nature de l'anomalie pour faciliter l'analyse par l'opérateur réseau et construire une signature de l'anomalie par combinaison de règles. La signature ainsi produite peut ensuite être utilisée pour détecter l'anomalie via un très classique système de détection par signature. Pour produire les règles de filtrage, l'algorithme conserve les sous-espaces $\mathbf{X}_n$ dans lesquels le ou les flux anormaux sont éloignés du reste du trafic. Nous définissons deux classes de filtrage différentes : les règles absolues notées $AFR(\mathbf{Y})$ et les règles relatives notées $RFR(\mathbf{Y})$. Les règles absolues ne dépendent pas de la séparation entre *clusters*, et correspondent à la présence de valeurs dominantes pour les attributs des flux anormaux. Une règle absolue sur l'attribut a qui caractérise un ensemble de flux $\mathbf{Y}_g \subset \mathbf{Y}$ avec $\mathbf{Y}_g$ contenant une anomalie est de la forme $AFR_a(\mathbf{Y}_g) = \{\forall \mathbf{y}_f \in \mathbf{Y}_g \subset \mathbf{Y} : x_f(a) == \lambda\}$. Par exemple, dans le cas d'une attaque ICMP flooding, la majorité des flux comportent uniquement des paquets ICMP. La règle de filtrage absolue $\{\text{nICMP/nPkts} == 1\}$ s'applique. Les règles de filtrage relatives dépendent de la séparation entre les flux normaux et anormaux. Si les flux anormaux sont séparés du reste du trafic dans un résultat de partitionnement P_n, alors les attributs du sous-espace correspondant $\mathbf{X}_n$ peuvent être utilisés pour définir une règle relative. Une règle relative sur l'attribut a qui caractérise un ensemble de flux $\mathbf{Y}_g \subset \mathbf{Y}$ avec $\mathbf{Y}_g$ contenant une anomalie est de la forme $RFR_a(\mathbf{Y}_g) = \{\forall \mathbf{y}_f \in \mathbf{Y}_g \subset \mathbf{Y} : x_f(a) < \lambda || x_f(a) > \lambda\}$. Une relation de couverture entre deux règles relatives est définie dans le cas où deux règles énoncent une relation sur le même attribut et selon le même position par rapport au seuil (inférieure ou supérieure). E.g. : soient les règle $f_a(\mathbf{Y}_g) = \{\forall \mathbf{y}_f \in \mathbf{Y}_g \subset \mathbf{Y} : x_f(a) < \lambda_1\}$ et $f_a(\mathbf{Y}_g) = \{\forall \mathbf{y}_f \in \mathbf{Y}_g \subset \mathbf{Y} : x_f(a) < \lambda_2\}$, la règle $f_a(\mathbf{Y}_g)$ couvre la règle $f_a(\mathbf{Y}_g)$ si et seulement si $\lambda_1 < \lambda_2$. Si deux règles ou plus se couvrent, l'algorithme conserve celle qui couvre les autres. Pour construire une signature concise de l'anomalie, la règle de filtrage la plus discriminante est choisie. Les règles absolues sont importantes, car elles définissent des caractéristiques inhérentes aux anomalies. Les règles relatives ont un intérêt qui dépend du degré de séparation des flux anormaux entre eux. Dans le cas d'outliers, nous choisissons les K attributs pour lesquels la distance normalisée au trafic normal (représenté par le *cluster* le plus grand dans chaque sous-espace) est parmi les K distances les plus importantes. Dans le cas de *clusters* de petites tailles, le degré de séparation avec le reste des *clusters* est ordonné en utilisant le score de Fisher (FS) [**?**], et les K règles les mieux classées sont sélectionnées. Le score de Fisher mesure la séparation entre *clusters*, en fonction de la variance totale à l'intérieur de chaque *cluster*. Enfin, pour construire une signature, les règles absolues et les K principales règles relatives sont combinées en un seul prédicat, en utilisant la règle de couverture en cas de superposition. Cet ensemble de méthodes pour construire les règles et composer une signature permet au système de caractériser de façon efficace les anomalies détectées et de fournir une mise en forme simple, concise et facile à comprendre.

D.4 Résumé

Ce chapitre présente plusieurs techniques que nous avons développées dans le but de traiter les anomalies après leur découverte. Nous réduisons tout d'abord le nombre d'anomalies en corrélant les anomalies issues de différents niveaux d'agrégations l_i. Nous réutilisons ce résultat et deux autres caractéristiques des anomalies pour construire un indice de dangerosité qui nous permet de hiérarchiser les anomalies. Nous caractérisons enfin les anomalies via une méthode qui construit une signature détaillée de chaque anomalie qui permette à l'opérateur de facilement comprendre la nature de l'anomalie. Cet ensemble de techniques facilite et réduit le travail de l'opérateur.

Ces différentes techniques sont partiellement détaillées dans le chapitre E à travers un cas pratique où UNADA est appliqué à du trafic réseau réel.

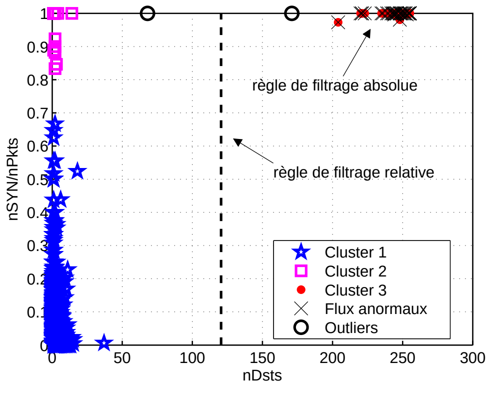
Chapitre E

Evaluation

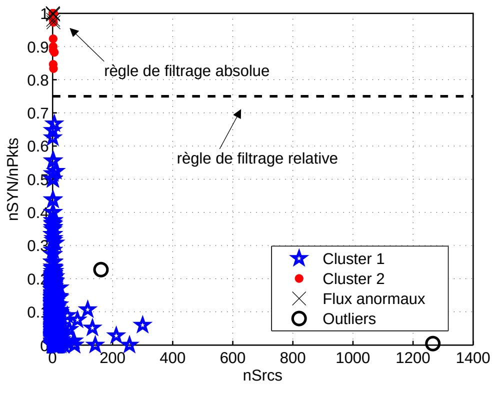
Nous évaluons ici la capacité de notre algorithme non supervisé à détecter et caractériser plusieurs anomalies contenues dans des traces de trafic réel de la base de traces MAWI du projet WIDE [4]. Le réseau opérationnel WIDE interconnecte des institutions de recherche au Japon, des fournisseurs d'accès internet et des universités aux Etats-Unis. Cette base contient des traces de paquets de 15 minutes collectées depuis 1999 et documentées de manière parcellaire [83].

E.1 Cas d'étude : une fenêtre temporelle dans une trace de la base de trace MAWI

Tout d'abord, nous avons analysé manuellement une trace WIDE choisie arbitrairement. Nous avons détecté et caractérisé un scan réseau de type SYN en direction de plusieurs machines appartenant au même réseau /16. Les paquets sont agrégés en flux IPdst/24 dans $\mathbf{Y}$. La longueur de la fenêtre temporelle est ΔT=15 secondes.



(a) SYN Network Scan (1/2)



(b) SYN Network Scan (2/2)

FIGURE E.1 – Règles de filtrage pour la caractérisation des attaques de la trace WIDE.

Si on s'intéresse aux règles de filtrage et à la signature associée à l'anomalie, les figures E.1a et E.1b montrent certains résultats de *clustering* P_n pour lesquels les règles absolues et les K règles relatives ont été produites. L'anomalie est ici représentée par un *cluster*. Les règles sont construites sur les attributs suivants : nombre de sources et de destinations et proportion de paquets SYN. En les combinant, on obtient la signature (nSrcs == 1) $\wedge$ (nDsts $> \lambda_1$) $\wedge$ (nSYN/nPkts $> \lambda_2$) où λ_1 et λ_2 sont deux seuils obtenus en séparant les *clusters* à mi distance de la moyenne de chaque *cluster* pour l'attribut considéré. L'utilisation de la moyenne permet d'avoir une approximation des valeurs du *cluster* pour l'attribut en question. Cette signature et le résultat du partitionnement sont parfaitement cohérents avec ce qu'annonce le tableau C.2 : le scan réseau forme un *cluster*, il émet des paquets d'une source unique vers un grand nombre de cibles et la majorité des paquets sont de type SYN. L'immense nouveauté avec cette approche se situe au niveau de la génération de cette signature sans aucune connaissance préalable sur l'anomalie et le trafic de fond.

E.2 Évaluation des performance d'UNADA

Cette section aborde l'évaluation de notre algorithme sur des traces de trafic réel extraites de la base de trace MAWI. Cette étape est critique dans la mesure où elle permet de mesurer l'efficacité de notre algorithme.

E.2.1 MAWILab : une documentation pour les anomalies contenues dans MAWI

Nous considérons les données de MAWILab [83] comme référence pour l'évaluation de notre travail. Dans [83], Fontugne et al. combinent quatre algorithmes de détection des anomalies du trafic via l'algorithme SCANN afin d'obtenir un seul ensemble d'anomalies. Les algortihmes de détection utilisés sont : un algorithme basé sur la *PCA* et les *sketchs* [97, 105], un algorithme basé sur les *sketchs* et la modélisation gamma multi-résolution [92], un algorithme basé sur la transformée de Hough [143] et un algorithme basé sur la détection de changement sur les distributions d'attributs du trafic [93].

Le site MAWILab [1] offre un accès aux résultats de la méthode sur le trafic de la base de trace MAWI sur les points de mesure B (de décembre 2000 à juin 2006) et F (d'août 2006 à aujourd'hui). Chaque fichier est au format XML et a été générée via la librairie admd [2].

Les auteurs de MAWILab fournissent également un outil appelé "MAWILab_Evaluate" qui compare un fichier XML de référence avec un autre fichier contenant le résultat d'un algorithme à évaluer. MAWILab_Evaluate fournit les quatre résultats classiques de test statistique : vrai positif (VP), faux positif (FP), faux négatif (FN) et vrai négatif (VN). Cet outil fournit ces quatre valeurs pour trois cas différents : *event* ou événements, *flows* ou flux et *alarms* ou alarmes. Les résultats de tests statistiques pour les flux considèrent chaque flux de manière atomique. Par contre, le taux de vrais positifs pour les événements représentent des anomalies pour lesquelles au moins un flux a été détecté comme étant anormal. Le nombre d'évènements vrais positifs représente donc un nombre de flux appartenant à des anomalies dont au moins un des flux est considéré comme anormal. Ceci est aussi valable pour les alarmes. À la différence près que les nombre d'alarmes vraies

[1] http ://www.fukuda-lab.org/mawilab/

[2] http ://admd.sourceforge.net/

positives représentent un nombre d'alarmes et non de flux. Par définition, il n'existe pas de vrai négatif pour les alarmes.

E.2.2 Résultats préliminaires

Nous procédons à une évaluation préliminaire sur la base de trace MAWI en utilisant une trace par année de 2001 à 2006. Notre algorithme présente un taux de faux positifs (FP) qui reste constamment bas tout au long des expérimentations de cette section. UNADA détecte seulement 12% des flux. Il détecte cependant 95% des événements et 89% des alarmes. Nous démontrons ensuite que si l'on enlève l'étape de détection de changement (cf. section C.1) afin de lancer la détection non-supervisée à chaque fenêtre temporelle, les performances s'améliorent. UNADA est désormais capable de détecter 45% des flux anormaux, 97% des événements et 98% des alarmes.

E.2.3 Analyse de sensibilité

Nous menons une analyse de sensibilité d'UNADA envers les paramètres suivants : epsDBS, nbpDBS, tICSim et tPropPtsNormC.

epsDBS est un paramètre de l'algorithme de partitionnement DBSCAN qui définit le voisinage d'un point. L'analyse de résultats montre que $epsDBS = 0.15$ offre les meilleures performances pour les trois critères offerts par MAWILab (événements, flux et alarmes). $epsDBS = 0.1$ offre de mauvaises performances en termes d'événements et des performances moyennes en termes d'alarmes. $epsDBS = 0.2$ offre des performances moyennes pour les événements mais quelque peu dégradées pour les alarmes. $epsDBS = 0.3$ offre de mauvaise performances pour les événements et alarmes . Les performances de chacun de ces réglages sont proches en termes de flux.

$nbpDBS$ est l'autre paramètre de DBSCAN qui définit le voisinage. Les valeurs testées mettent en évidence une très faible sensibilité d'UNADA envers ce paramètre. Les raisons de cette faible sensibilité sont au nombre de deux. Premièrement, les valeurs testées n'empêchent pas l'apparition d'un *cluster* représentant le trafic normal (ce qui dégraderait les performances en bloquant l'analyse de certaines fenêtres temporelles, cf. hypothèses C.3 et C.4). Enfin, MAWILab_Evaluate a un point de vue global sur les anomalies identifiées. Il ne tient pas compte des éventuelles inclusions et/ou complémentarités entre éléments détectées. nbpDBS déterminant la représentation d'une anomalie, soit un *cluster* soit plusieurs *outliers* (cf. section C.2.1), il n'a pas d'influences sur les performances du système.

$tICSim$ est le seuil appliqué au critère de similarité entre *cluster* utilisé pour ICRA. Les valeurs $tICSim = 0.7$, $tICSim = 0.85$ et $tICSim = 0.95$ produisent des résultats très proches. $tICSim = 0.5$ produit des résultats légèrement inférieurs. La sensibilité d'UNADA envers $tICSim$ est cependant globalement faible.

$tPropPtsNormC$, noté aussi Y, représente le seuil appliqué à la proportion de trafic normal (cf. hypothèses C.3 et C.4). Nos analyses démontrent que les valeurs 0.5, 0.6 et 0.8 donnent de bonnes performances. Les valeurs 0.85 et 0.95 présentent des performances dégradées. UNADA présente donc de bonnes performances pour une large plage de réglages possible.

E.2.4 Analyse globale

Afin d'avoir une idée plus précise quant à l'efficacité de notre algorithme, nous utilisons un sous-ensemble plus important de la base de trace MAWI. Nous utilisons ainsi 4 traces par mois de 2001 à 2006. Les valeurs de paramètres utilisés sont les suivantes : epsDBS = 0.15, nbpDBS = 4, tICSim = 0.85 and tPropPtsNormC (or Y) = 0.6.

Les résultats montrent qu'UNADA détecte une faible proportion de flux anormaux. UNADA est cependant capable de détecter plus de 95% des événements anormaux.

E.3 Résumé

Ce chapitre présente l'évaluation d'UNADA. Cette évaluation est réalisée en utilisant les données fournies par le projet MAWILab comme référence. La sensibilité d'UNADA est faible vis-à-vis des paramètres $nbpDBS$ et $tICSim$. Cette sensibilité est plus importante vis-à-vis des paramètres $epsDBS$ et $tPropPtsNormC$. L'étendue de la plage de réglages donnant de bons résultats est suffisamment grande pour permettre de facilement régler UNADA.

Notre algorithme présente un faible taux de faux positifs. Ceci est très intéressant dans la mesure où cela garantit que l'opérateur ne sera pas submergé par les alarmes. Notre algorithme est de plus capable de détecter une écrasante majorité des événements anormaux répertoriés par MAWILab.

Chapitre F

Conclusion

Notre algorithme non supervisé de détection et de caractérisation d'anomalies supplante les méthodes précédemment proposées dans ce domaine, et ce, sans nécessiter de connaissance préalable des anomalies et du trafic de fond. L'algorithme proposé parvient, grâce aux techniques de partitionnement proposées d'extraire les flux anormaux. Il est aussi capable de corréler, hiérarchiser et caractériser les anomalies préalablement extraites. Ces signatures peuvent alors être traduites dans le langage des équipements de sécurité (systèmes de détection d'intrusion, firewalls, ...) et y être utilisées automatiquement. Notre algorithme peut donc être implanté à l'intérieur de systèmes de sécurité autonomes dans lequel la détection/caractérisation non-supervisée fonctionne en parallèle avec un système à base de signature afin d'identifier les événements anormaux inconnus et être capable de les détecter immédiatement après.

Les perspectives de notre travail sont vastes. Une première amélioration consisterait à utiliser un plus grand nombre de niveaux d'agrégation afin d'améliorer la phase de corrélation. Les techniques de classement d'anomalies pourraient aussi être améliorées en réutilisant les résultats de partitionnement et essayant d'incorporer l'éloignement au trafic normal à l'indice de dangerosité. L'étape de caractérisation pourrait elle être améliorée via l'utilisation d'algorithmes d'apprentissage supervisé (tels que les arbres de décisions) en lieu et place des heuristiques actuelles.

Notre travail nous semble donc représenter une première étape déterminante vers l'introduction de systèmes de détection d'anomalies autonomes qui peuvent ainsi parfaitement s'intégrer à des systèmes de gestion des réseaux autonomes.

Acronyms

ADSL Asymmetric Digital Subscriber Line. 6, 9

AMR Adaptive Mesh Refinement. 20

ARP Address Resolution Protocol. 30, 31

ART Adaptive Resonance Theory. 17

AS Autonomous System. 5, 26

BGP Border Gateway Protocol. 26

BSS Blind Signal Separation. 16

DDoS Distributed Denial of Service. 4, 9, 30, 38

DoS Denial of Service. 4, 9, 38

EA4C EA for small-Clusters identification. 43, 45

EA4O EA for Outliers identification. 43

EAC Evidence Accumulation Clustering. 43, 45

FFT Fast Fourier Transform. 28

FPR False Positive Rate. 45, 68

FTTH Fiber-To-The-Home. 6

GA Genetic Algorithms. 30

GPU Graphic Processor Unit. 49

HTTP Hypertext Transfer Protocol. 9

ICA Independent Component Analysis. 16, 40

ICLA Inter-CLuster Association. 46, 61, 65, 75

ICMP Internet Control Message Protocol. 30, 58

ICRA Inter-Clustering Results Association. 49, 61, 75

IDS Intrusion Detection Systems. 13, 25

IOA Inter-Outlier Association. 46, 63, 65, 75

IP Internet Protocol. 26–28, 30, 34–39, 47, 55, 61, 64, 65, 67, 70, 81, 82

IPS Intrusion Protection Systems. 13, 25

ISP Internet Service Provider. 4, 5, 13, 60

ISPs Internet Service Providers. 5

LBNL Lawrence Berkeley National Laboratory. 10

NNTP Network News Transfer Protocol. 10

OD-flows Origin-Destination flows. 28, 29

PCA Principal Component Analysis. 16, 17, 23, 24, 27–29, 40, 67

PoP Point of Presence. 6

PoPs Points of Presence. 6, 8

ROC Receiver operating characteristic. 66

SNMP Simple Network Management Protocol. 5, 26

SOFM Self-Organizing Feature Map. 30

SOM Self-Organizing Maps. 17

SSC Sub-Space Clustering. 41, 43, 49

SSH Secure Shell. 10

SVM Support Vector Machine. 30, 31

TCP Transmission Control Protocol. 30

TPR True Positive Rate. 45, 68

UDP User Datagram Protocol. 30

VoD Video on Demand. 8

Bibliography

[1] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang, "Experience in measuring backbone traffic variability: models, metrics, measurements and meaning," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, IMW '02, (New York, NY, USA), pp. 91–92, ACM, 2002.

[2] K. Papagiannaki, N. Taft, Z.-L. Zhang, and C. Diot, "Long-term forecasting of internet backbone traffic: observations and initial models," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 2, pp. 1178 – 1188 vol.2, march-3 april 2003.

[3] K. Cho, K. Fukuda, H. Esaki, and A. Kato, "The impact and implications of the growth in residential user-to-user traffic," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06, (New York, NY, USA), pp. 207–218, ACM, 2006.

[4] K. Cho, K. Mitsuya, and A. Kato, "Traffic data repository at the wide project," in *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '00, (Berkeley, CA, USA), pp. 51–51, USENIX Association, 2000.

[5] K. Cho, K. Fukuda, H. Esaki, and A. Kato, "Observing slow crustal movement in residential user traffic," in *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, (New York, NY, USA), pp. 12:1–12:12, ACM, 2008.

[6] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2010–2015," tech. rep., 6 2010.

[7] H. Ringberg, A. Soule, J. Rexford, and C. Diot, "Sensitivity of pca for traffic anomaly detection," in *Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '07, (New York, NY, USA), pp. 109–120, ACM, 2007.

[8] Sandvine, "Global internet phenomena report: Fall 2011," 2011. `http://www.sandvine.com/downloads/documents/2010%20Global%20Internet%20Phenomena%20Report.pdf`.

[9] Sandvine, "Global internet phenomena spotlight: Netflix rising," 2011. `http://www.sandvine.com/downloads/documents/05-17-2011_phenomena/Sandvine%20Global%20Internet%20Phenomena%20Spotlight%20-%20Netflix%20Rising.pdf`.

[10] D. Plonka, "Uw-madison napster traffic measurement," 2000. http://net.doit.wisc.edu/data/Napster/.

[11] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos, "Is p2p dying or just hiding? [p2p traffic measurement]," in *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, vol. 3, pp. 1532 – 1538 Vol.3, nov.-3 dec. 2004.

[12] P. Borgnat, G. Dewaele, K. Fukuda, P. Abry, and K. Cho, "Seven years and one day: Sketching the evolution of internet traffic," in *INFOCOM 2009, IEEE*, pp. 711 –719, april 2009.

[13] K. Cho, *Broadband Traffic Report, Traffic Shifting away from P2P File Sharing to Web Services*, pp. 25–30. Internet Initiative, Japan, 2010.

[14] G. Maier, A. Feldmann, V. Paxson, and M. Allman, "On dominant characteristics of residential broadband internet traffic," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, (New York, NY, USA), pp. 90–102, ACM, 2009.

[15] Cisco, "Cisco Visual Networking Index: Usage," tech. rep., 10 2010.

[16] M. Allman, V. Paxson, and J. Terrell, "A brief history of scanning," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, (New York, NY, USA), pp. 77–82, ACM, 2007.

[17] J. D. Brutlag, "Aberrant behavior detection in time series for network monitoring," in *Proceedings of the 14th USENIX conference on System administration*, pp. 139–146, 2000.

[18] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *Proceedings of the ACM SIGCOMM 2005 conference*, SIGCOMM '05, (New York, NY, USA), pp. 217–228, ACM, 2005.

[19] T. Shon and J. Moon, "A hybrid machine learning approach to network anomaly detection," *Inf. Sci.*, vol. 177, pp. 3799–3821, September 2007.

[20] N. Duffield, P. Haffner, B. Krishnamurthy, and H. Ringberg, "Rule-based anomaly detection on ip flows," in *INFOCOM 2009, IEEE*, pp. 424 –432, april 2009.

[21] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. 13, pp. 222–232, feb. 1987.

[22] H. S. Javitz, A. Valdes, A. S. Ooie, and R. D. Patton, "The nides statistical component: Description and justification," tech. rep., 1994.

[23] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion detection with unlabeled data using clustering," in *Proceedings of ACM CSS Workshop on Data Mining Applied to Security*, DMSA-2001, pp. 5–8, 2001.

[24] I. Jolliffe, *Principal component analysis.* Springer series in statistics, Springer-Verlag, 2002.

[25] K. Pearson, “On lines and planes of closest fit to systems of points in space,” *Philosophical Magazine*, vol. 2, no. 6, pp. 559–572, 1901.

[26] J. Vesanto and E. Alhoniemi, “Clustering of the self-organizing map,” *Neural Networks, IEEE Transactions on*, vol. 11, pp. 586 –600, may 2000.

[27] B. Mazzotta, “Prominence and polarity in international trade,” in *Proceedings of the annual meeting of the International Studies Association Annual Conference ”Global Governance: Political Authority in Transition”*, 03 2011.

[28] J. B. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* (L. M. L. Cam and J. Neyman, eds.), vol. 1, pp. 281–297, University of California Press, 1967.

[29] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proc. of 2nd International Conference on Knowledge Discovery and Data Mining*, KDD’1996, pp. 226–231, AAAI Press, 1996.

[30] H.-P. Kriegel, P. Kröger, and A. Zimek, “Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering,” *ACM Trans. Knowl. Discov. Data*, vol. 3, pp. 1:1–1:58, March 2009.

[31] L. Parsons, E. Haque, and H. Liu, “Subspace clustering for high dimensional data: a review,” *SIGKDD Explor. Newsl.*, vol. 6, pp. 90–105, June 2004.

[32] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, “Fast algorithms for projected clustering,” in *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, SIGMOD ’99, (New York, NY, USA), pp. 61–72, ACM, 1999.

[33] K.-G. Woo, J.-H. Lee, M.-H. Kim, and Y.-J. Lee, “Findit: a fast and intelligent subspace clustering algorithm using dimension voting,” *Information and Software Technology*, vol. 46, no. 4, pp. 255 – 271, 2004.

[34] K. Y.-L. Yip, D. W. Cheung, and M. K. Ng, “On discovery of extremely low-dimensional clusters using semi-supervised projected clustering,” in *Proceedings of the 21st International Conference on Data Engineering*, ICDE ’05, (Washington, DC, USA), pp. 329–340, IEEE Computer Society, 2005.

[35] C. Böhm, K. Kailing, H.-P. Kriegel, and P. Kröger, “Density connected clustering with local subspace preferences,” in *Proceedings of the Fourth IEEE International Conference on Data Mining*, ICDM ’04, (Washington, DC, USA), pp. 27–34, IEEE Computer Society, 2004.

[36] C. Domeniconi, D. Papadopoulos, D. Gunopulos, and S. Ma, "Subspace clustering of high dimensional data," in *Proceedings of the Fourth SIAM International Conference on Data Mining*, SIAM, April 2004.

[37] J. H. Friedman and J. J. Meulman, "Clustering objects on subsets of attributes," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 66, no. 4, pp. 815–849, 2004.

[38] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," SIGMOD '98, pp. 94–105.

[39] H. Nagesh, S. Goil, and A. Choudhary, "Adaptive grids for clustering massive data sets," in *In 1st SIAM International Conference Proceedings on Data Mining*, 2001.

[40] C.-H. Cheng, A. W. Fu, and Y. Zhang, "Entropy-based subspace clustering for mining numerical data," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '99, (New York, NY, USA), pp. 84–93, ACM, 1999.

[41] K. Kailing, H.-P. Kriegel, and P. Kröger, "Density-connected subspace clustering for high-dimensional data," in *Proceedings of the 4th SIAM International Conference on Data Mining (SDM)*, pp. 246–257, 2004.

[42] I. Assent, R. Krieger, E. Müller, and T. Seidl, "Dusc: Dimensionality unbiased subspace clustering," in *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, (Washington, DC, USA), pp. 409–414, IEEE Computer Society, 2007.

[43] W.-k. Liao, Y. Liu, and A. Choudhary, "A grid-based clustering algorithm using adaptive mesh refinement," 2004.

[44] M.-I. Akodjènou-Jeannin, K. Salamatian, and P. Gallinari, "Flexible grid-based clustering," in *Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases*, PKDD 2007, (Berlin, Heidelberg), pp. 350–357, Springer-Verlag, 2007.

[45] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali, "A monte carlo algorithm for fast projective clustering," in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, SIGMOD '02, (New York, NY, USA), pp. 418–427, ACM, 2002.

[46] M. L. Yiu and N. Mamoulis, "Frequent-pattern based iterative projected clustering," in *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, (Washington, DC, USA), pp. 689–, IEEE Computer Society, 2003.

[47] M. L. Yiu and N. Mamoulis, "Iterative projected clustering by subspace mining," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, pp. 176 – 189, feb. 2005.

[48] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, I. Müller-Gorman, and A. Zimek, "Detection and visualization of subspace cluster hierarchies," in *Proceedings of the 12th international conference on Database systems for advanced applications*, DASFAA'07, (Berlin, Heidelberg), pp. 152–163, Springer-Verlag, 2007.

[49] K. Y.-L. Yip, D. Cheung, and M. Ng, "Harp: a practical projected clustering algorithm," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 16, pp. 1387 – 1397, nov. 2004.

[50] K. Sequeira and M. Zaki, "Schism&#58; a new approach to interesting subspace mining," *Int. J. Bus. Intell. Data Min.*, vol. 1, pp. 137–160, December 2005.

[51] H.-P. Kriegel, P. Kröger, M. Renz, and S. Wurst, "A generic framework for efficient subspace clustering of high-dimensional data," in *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, (Washington, DC, USA), pp. 250–257, IEEE Computer Society, 2005.

[52] G. Moise, J. Sander, and M. Ester, "P3c: A robust projected clustering algorithm," in *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, (Washington, DC, USA), pp. 414–425, IEEE Computer Society, 2006.

[53] G. Moise, J. Sander, and M. Ester, "Robust projected clustering," *Knowl. Inf. Syst.*, vol. 14, pp. 273–298, March 2008.

[54] C. C. Aggarwal and P. S. Yu, "Finding generalized projected clusters in high dimensional spaces," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, SIGMOD '00, (New York, NY, USA), pp. 70–81, ACM, 2000.

[55] C. Böhm, K. Kailing, P. Kröger, and A. Zimek, "Computing clusters of correlation connected objects," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, (New York, NY, USA), pp. 455–466, ACM, 2004.

[56] E. Achtert, C. Böhm, P. Kröger, and A. Zimek, "Mining hierarchies of correlation clusters," in *Scientific and Statistical Database Management, 2006. 18th International Conference on*, pp. 119 –128, 0-0 2006.

[57] E. Achtert, C. Böhm, J. David, P. Kröger, and A. Zimek, "Robust clustering in arbitrarily oriented subspaces.," in *Proceedings of the 8th SIAM International Conference on Data Mining (SDM)*, pp. 763–774, SIAM, 2008.

[58] D. Barbarà and P. Chen, "Using the fractal dimension to cluster datasets," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, (New York, NY, USA), pp. 260–264, ACM, 2000.

[59] E. P. M. Sousa, C. Traina Jr., A. J. M. Traina, and C. Faloutsos, "How to use fractal dimension to find correlations between attributes," in *First Workshop on Fractals and Self-similarity in Data Mining: Issues and Approaches (in conjunction with*

8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining), (Edmonton, Alberta, Canada), pp. 26–30, ACM Press, ACM Press, 2002.

[60] A. Gionis, A. Hinneburg, S. Papadimitriou, and P. Tsaparas, "Dimension induced clustering," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, (New York, NY, USA), pp. 51–60, ACM, 2005.

[61] R. Haralick and R. Harpaz, "Linear manifold clustering," in *Proceedings of the 4th International Conference on Machine Learning and Data Mining in Pattern Recognition* (P. Perner and A. Imiya, eds.), vol. 3587 of *Lecture Notes in Computer Science*, pp. 641–641, Springer Berlin / Heidelberg, 2005. 10.1007/11510888_14.

[62] R. Harpaz and R. Haralick, "Mining subspace correlations," in *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining*, CIDM 2007, pp. 335 –342, april 2007.

[63] J. A. Hartigan, "Direct clustering of a data matrix," *Journal of the American Statistical Association*, vol. 67, no. 337, pp. 123–129, 1972.

[64] A. Califano, G. Stolovitzky, and Y. Tu, "Analysis of gene expression microarrays for phenotype classification," in *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pp. 75–85, AAAI Press, 2000.

[65] Q. Sheng, Y. Moreau, and B. De Moor, "Biclustering microarray data by gibbs sampling," *Bioinformatics*, no. 337, pp. 196–205, 2003.

[66] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller, "Rich probabilistic models for gene expression," vol. 17, pp. 243–252, 2001.

[67] Y. Cheng and G. M. Church, "Biclustering of expression data," in *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pp. 93–103, AAAI Press, 2000.

[68] J. Yang, W. Wang, H. Wang, and P. Yu, "d-clusters: Capturing subspace correlation in a large data set," in *Proceedings of the 18th International Conference on Data Engineering*, ICDE '02, (Washington, DC, USA), pp. 517–, IEEE Computer Society, 2002.

[69] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra, "Minimum sum squared residue co-clustering of gene expression data," in *Proceedings of the 4th SIAM International Conference on Data Mining (SDM)*, pp. 114–125, SIAM, 2004.

[70] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini, "Discovering local structure in gene expression data: the order-preserving submatrix problem," in *Proceedings of the sixth annual international conference on Computational biology*, RECOMB '02, (New York, NY, USA), pp. 49–57, ACM, 2002.

[71] J. Liu and W. Wang, "Op-cluster: Clustering by tendency in high dimensional space," in *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, (Washington, DC, USA), pp. 187–, IEEE Computer Society, 2003.

[72] J.-W. Chang and D.-S. Jin, "A new cell-based clustering method for large, high-dimensional data in data mining applications," in *Proceedings of the 2002 ACM symposium on Applied computing*, SAC '02, (New York, NY, USA), pp. 503–507, ACM, 2002.

[73] B. Liu, Y. Xia, and P. S. Yu, "Clustering through decision tree construction," in *Proceedings of the ninth international conference on Information and knowledge management*, CIKM '00, (New York, NY, USA), pp. 20–29, ACM, 2000.

[74] E. Müller, S. Günnemann, I. Assent, and T. Seidl, "Evaluating clustering in subspace projections of high dimensional data," *Proc. VLDB Endow.*, vol. 2, pp. 1270–1281, August 2009.

[75] I. Assent, R. Krieger, E. Müller, and T. Seidl, "Inscy: Indexing subspace clusters with in-process-removal of redundancy," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, (Washington, DC, USA), pp. 719–724, IEEE Computer Society, 2008.

[76] G. Moise and J. Sander, "Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering," in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, (New York, NY, USA), pp. 533–541, ACM, 2008.

[77] G. Moise, A. Zimek, P. Kröger, H.-P. Kriegel, and J. Sander, "Subspace and projected clustering: experimental evaluation and analysis," *Knowl. Inf. Syst.*, vol. 21, pp. 299–326, November 2009.

[78] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[79] K. Y. Yip, P. Qi, M. H. Schultz, D. W. Cheung, and K.-H. Cheung, "Sembiosphere: A semantic web approach to recommending microarray clustering services," in *Pacific Symposium on Biocomputing*, pp. 188–199, 2006.

[80] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data An Introduction to Cluster Analysis*. New York: Wiley Interscience, 1990.

[81] R. T. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," in *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, (San Francisco, CA, USA), pp. 144–155, Morgan Kaufmann Publishers Inc., 1994.

[82] A. B. Ashfaq, M. Javed, S. A. Khayam, and H. Radha, "An information-theoretic combining method for multi-classifier anomaly detection systems," in *IEEE International Conference on Communications*, ICC '10, pp. 1 –5, may 2010.

[83] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "Mawilab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in *Proceedings of the 6th International Conference on emerging Networking EXperiments and Technologies*, Co-NEXT '10, (New York, NY, USA), pp. 1–12, ACM, 2010.

[84] S. Shanbhag and T. Wolf, "Accurate anomaly detection through parallelism," *IEEE Network*, vol. 23, pp. 22 –28, January 2009.

[85] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer Networks*, vol. 31, pp. 2435–2463, December 1999.

[86] F. Silveira and C. Diot, "Urca: pulling out anomalies by their root causes," in *Proceedings of the 29th IEEE International Conference on Computer Communications*, INFOCOM'10, (Piscataway, NJ, USA), pp. 722–730, IEEE Press, march 2010.

[87] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan, "Network anomography," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, IMC '05, (Berkeley, CA, USA), pp. 30–30, USENIX Association, 2005.

[88] G. Cormode and S. M. Muthukrishnan, "What's new: finding significant differences in network data streams," *Networking, IEEE/ACM Transactions on*, vol. 13, pp. 1219 – 1232, Dec. 2005.

[89] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: methods, evaluation, and applications," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, IMC '03, (New York, NY, USA), pp. 234–247, ACM, 2003.

[90] M. Roughan, T. Griffin, M. Mao, A. Greenberg, and B. Freeman, "Combining routing and traffic data for detection of ip forwarding anomalies," in *Proceedings of the joint international conference on Measurement and modeling of computer systems*, SIGMETRICS '04/Performance '04, (New York, NY, USA), pp. 416–417, ACM, 2004.

[91] A. Soule, K. Salamatian, and N. Taft, "Combining filtering and statistical methods for anomaly detection," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, IMC '05, (Berkeley, CA, USA), pp. 331–344, USENIX Association, 2005.

[92] G. Dewaele, K. Fukuda, P. Borgnat, P. Abry, and K. Cho, "Extracting hidden anomalies using sketch and non gaussian multiresolution statistical detection procedures," LSAD '07, pp. 145–152.

[93] D. Brauckhoff, X. Dimitropoulos, A. Wagner, and K. Salamatian, "Anomaly extraction in backbone networks using association rules," in *Proceedings of the 9th ACM SIGCOMM conference on Internet Measurement Conference*, IMC '09, (New York, NY, USA), pp. 28–34, ACM, 2009.

[94] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules in large databases,” in *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB ’94, (San Francisco, CA, USA), pp. 487–499, Morgan Kaufmann Publishers Inc., 1994.

[95] P. Barford, J. Kline, D. Plonka, and A. Ron, “A signal analysis of network traffic anomalies,” in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, IMW ’02, (New York, NY, USA), pp. 71–82, ACM, 2002.

[96] D. Brauckhoff, K. Salamatian, and M. May, “Applying pca for traffic anomaly detection: Problems and solutions,” in *Proceedings of the 28th IEEE International Conference on Computer Communications*, pp. 2866 –2870, april 2009.

[97] Y. Kanda, K. Fukuda, and T. Sugawara, “Evaluation of anomaly detection based on sketch and pca,” in *GLOBECOM 2010, 2010 IEEE Global Telecommunications Conference*, pp. 1 –5, dec. 2010.

[98] P. Barford and D. Plonka, “Characteristics of network traffic flow anomalies,” in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, IMW ’01, (New York, NY, USA), pp. 69–73, ACM, 2001.

[99] F. Silveira, C. Diot, N. Taft, and R. Govindan, “Astute: detecting a different class of traffic anomalies,” in *Proceedings of the ACM SIGCOMM 2010 conference*, SIGCOMM ’10, (New York, NY, USA), pp. 267–278, ACM, 2010.

[100] D. J. Daley and D. Vere-Jones, *An Introduction to the Theory of Point Processes: General theory and structure.* Probability and its applications, Springer, 2007.

[101] C. Barakat, P. Thiran, G. Iannaccone, C. Diot, and P. Owezarski, “A flow-based model for internet backbone traffic,” in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, IMW ’02, (New York, NY, USA), pp. 35–47, ACM, 2002.

[102] N. H. Darryl, D. Veitch, and P. Abry, “Cluster processes, a natural language for network traffic,” in *IEEE Transactions on Networking*, pp. 2229–2244, 2003.

[103] A. Lakhina, M. Crovella, and C. Diot, “Characterization of network-wide anomalies in traffic flows,” in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, IMC ’04, (New York, NY, USA), pp. 201–206, ACM, 2004.

[104] A. Lakhina, M. Crovella, and C. Diot, “Diagnosing network-wide traffic anomalies,” in *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM ’04, (New York, NY, USA), pp. 219–230, ACM, 2004.

[105] X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina, “Detection and identification of network anomalies using sketch subspaces,” in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC ’06, (New York, NY, USA), pp. 147–152, ACM, 2006.

[106] P. Chhabra, C. Scott, E. D. Kolaczyk, and M. Crovella, "Distributed spatial anomaly detection," in *Proceedings of the 27th IEEE International Conference on Computer Communication*, pp. 1705 –1713, april 2008.

[107] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-H. Lau, N. Taft, and J. D. Tygar, "Evading anomaly detection through variance injection attacks on pca," in *Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection*, RAID '08, (Berlin, Heidelberg), pp. 394–395, Springer-Verlag, 2008.

[108] B. I. P. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, N. Taft, and D. Tygar, "Compromising pca-based anomaly detectors for network-wide traffic," Tech. Rep. UCB/EECS-2008-73, EECS Department, University of California, Berkeley, May 2008.

[109] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. D. Tygar, "Antidote: understanding and defending against poisoning of anomaly detectors," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, (New York, NY, USA), pp. 1–14, ACM, 2009.

[110] A. Abdelkefi, Y. Jiang, W. Wang, A. Aslebo, and O. Kvittem, "Robust traffic anomaly detection with principal component pursuit," in *Proceedings of the ACM CoNEXT Student Workshop*, CoNEXT '10 Student Workshop, (New York, NY, USA), pp. 10:1–10:2, ACM, 2010.

[111] R. Sadoddin and A. A. Ghorbani, "A comparative study of unsupervised machine learning and data mining techniques for intrusion detection," in *Proceedings of the 5th international conference on Machine Learning and Data Mining in Pattern Recognition*, MLDM '07, (Berlin, Heidelberg), pp. 404–418, Springer-Verlag, 2007.

[112] J. Lei and A. Ghorbani, "Network intrusion detection using an improved competitive learning neural network," in *Communication Networks and Services Research, 2004. Proceedings. Second Annual Conference on*, pp. 190 – 197, may 2004.

[113] P. Gogoi, D. Bhattacharyya, B. Borah, and J. K. Kalita, "A survey of outlier detection methods in network anomaly identification," *Computer Journal*, vol. 54, pp. 570–588, April 2011.

[114] J. Zhang and M. Zulkernine, "Anomaly based network intrusion detection with unsupervised outlier detection," in *Communications, 2006. ICC '06. IEEE International Conference on*, vol. 5, pp. 2388 –2393, june 2006.

[115] KDD99, "Kdd99 cup dataset," 1999. `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

[116] Y. Yasami and S. P. Mozaffari, "A novel unsupervised classification approach for network anomaly detection by k-means clustering and id3 decision tree learning methods," *J. Supercomput.*, vol. 53, pp. 231–245, July 2010.

[117] L. Zi, J. Yearwood, and X.-W. Wu, "Adaptive clustering with feature ranking for ddos attacks detection," in *Proceedings of the 2010 Fourth International Conference on Network and System Security*, NSS '10, (Washington, DC, USA), pp. 281–286, IEEE Computer Society, 2010.

[118] L. Kuang and M. Zulkernine, "An anomaly intrusion detection method using the csi-knn algorithm," in *Proceedings of the 2008 ACM symposium on Applied computing*, SAC '08, (New York, NY, USA), pp. 921–926, ACM, 2008.

[119] X. Li and N. Ye, "Grid- and dummy-cluster-based learning of normal and intrusive clusters for computer intrusion detection," *Quality and Reliability Engineering International*, vol. 18, no. 3, pp. 231–242, 2002.

[120] N. Ye and X. Li, "A scalable clustering technique for intrusion signature recognition," pp. 1–4, 2001.

[121] K. Burbeck and S. Nadjm-tehrani, "Adwice - anomaly detection with real-time incremental clustering," in *Proceedings of the 7th International Conference on Information Security and Cryptology, Seoul, Korea*, Springer Verlag, 2004.

[122] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, SIGMOD '96, (New York, NY, USA), pp. 103–114, ACM, 1996.

[123] D. Barbará, J. Couto, S. Jajodia, L. Popyack, and N. Wu, "Adam: Detecting intrusions by data mining," in *Proceedings of the IEEE Workshop on Information Assurance and Security*, pp. 11–16, 2001.

[124] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data," in *Applications of Data Mining in Computer Security*, Kluwer, 2002.

[125] K. Leung and C. Leckie, "Unsupervised anomaly detection in network intrusion detection using clusters," in *Proceedings of the Twenty-eighth Australasian conference on Computer Science - Volume 38*, ACSC '05, (Darlinghurst, Australia, Australia), pp. 333–342, Australian Computer Society, Inc., 2005.

[126] J. Oldmeadow, S. Ravinutala, and C. Leckie, "Adaptive clustering for network intrusion detection," in *Advances in Knowledge Discovery and Data Mining* (H. Dai, R. Srikant, and C. Zhang, eds.), vol. 3056 of *Lecture Notes in Computer Science*, pp. 255–259, Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-24775-3_33.

[127] P. Casas, J. Mazel, and P. Owezarski, "Steps towards autonomous network security: Unsupervised detection of network attacks," in *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*, pp. 1 –5, feb. 2011.

[128] P. Casas, J. Mazel, and P. Owezarski, "Unada: unsupervised network anomaly detection using sub-space outliers ranking," in *Proceedings of the 10th international IFIP TC 6 conference on Networking - Volume Part I*, NETWORKING'11, (Berlin, Heidelberg), pp. 40–51, Springer-Verlag, 2011.

[129] P. Casas, J. Mazel, and P. Owezarski, "Knowledge-independent traffic monitoring: Unsupervised detection of network attacks," *IEEE Network, special issue on Network traffic monitoring and analysis, issue 2, to appear*, vol. 4, pp. 228–242, May/June 2012. Accepted for publication.

[130] J. Mazel, P. Casas, and P. Owezarski, "Sub-space clustering & evidence accumulation for unsupervised network anomaly detection," in *Proceedings of the Traffic Monitoring and Analysis Workshop*, TMA '11, April 2011.

[131] J. Mazel, P. Casas, Y. Labit, and P. Owezarski, "Sub-space clustering, inter-clustering results association & anomaly correlation for unsupervised network anomaly detection," in *7th International Conference on Network and Service Management (CNSM 2011)*, CNSM'11, October 2011.

[132] C. Barakat, A. Krifa, Y. Labit, I. Lassoued, J. Mazel, P. Owezarski, and K. Salamatian, "Deliverable d3.2 : implementation of adaptive traffic sampling and management, path performance monitoring and management, path performance monitoring and cooperative intrusion and attack/anomaly detection techniques," tech. rep., October 2009.

[133] A. Krifa, I. Lassoued, C. Barakat, P. Casas Hernandez, P. Owezarski, J. Mazel, and Y. Labit, "Deliverable d3.3. experimental evaluation of t01," tech. rep., November 2010.

[134] E. Achtert, H.-P. Kriegel, and A. Zimek, "Elki: A software system for evaluation of subspace clustering algorithms," in *Scientific and Statistical Database Management* (B. Ludäscher and N. Mamoulis, eds.), vol. 5069 of *Lecture Notes in Computer Science*, pp. 580–585, Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-69497-7_41.

[135] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, pp. 10–18, November 2009.

[136] Y. Minsky, "Ocaml for the masses," *Queue*, vol. 9, pp. 43:40–43:49, September 2011.

[137] A. Strehl and J. Ghosh, "Cluster ensembles — a knowledge reuse framework for combining multiple partitions," *Journal on Machine Learning Research*, vol. 3, pp. 583–617, March 2003.

[138] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[139] A. L. N. Fred and A. K. Jain, "Combining multiple clusterings using evidence accumulation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, pp. 835–850, June 2005.

[140] B. Immanuel, M. Budinich, P. M. Pardalos, and M. Pelillo, "The maximum clique problem," *Handbook of Combinatorial Optimization (supp. Vol. A)*, pp. 1–74.

[141] J.-C. Filliâtre and K. Kalyanasundaram, "Functory: A distributed computing library for objective caml," in *Trends in Functional Programming*, (Madrid, Spain), May 2011.

[142] P. Owezarski and G. Fernandes, "Automated classification of network traffic anomalies," in *Proceedings of the 7th International ICST Conference on Security and Privacy in Communication Networks*, SecurComm'09, pp. 91–100, 2009.

[143] R. Fontugne and K. Fukuda, "A hough-transform-based anomaly detector with an adaptive time interval," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, (New York, NY, USA), pp. 471–477, ACM, 2011.

[144] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, March 1986.

[145] J. R. Quinlan, *C4.5: programs for machine learning.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.