Addressing Practical Challenges for Anomaly Detection in Backbone Networks

Ignasi Paredes-Oliva

Advisor: Dr. Pere Barlet-Ros
Co-Advisors: Prof. Josep Solé-Pareta and
Dr. Xenofontas Dimitropoulos

A dissertation submitted in partial fulfillment of the requirements for the degree of:

Ph.D. in Computer Science

Universitat Politècnica de Catalunya BarcelonaTech
Department d'Arquitectura de Computadors

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Barcelona, June 2013

Acknowledgements

First of all I would like to show my gratitude to my advisor, Dr. Pere Barlet Ros for his help and guidance since I met him more than five years ago when I was still doing my masters degree. His knowledge and expertise have allowed me to successfully develop my thesis and improve significantly my technical and research skills. Also, I want to thank Pere for giving me the opportunity of doing an internship in Cambridge at the very beginning of this Ph.D. thesis.

I also wish to thank my co-advisors. I would like to show my gratitude to Prof. Josep Solé Pareta for giving me the chance to start working with the CCABA research group back in 2003 and for his continuous support, encouragement and advice. Also, I want to thank Dr. Xenofontas Dimitropoulos for allowing me to work with him in ETH Zürich and for his constructive discussions and ideas during the development of this thesis.

I would like to show my most honest gratitude to Maurizio Molina for his invaluable help and hospitality during my internship in the United Kingdom and also for his determinant role in the evolution of the thesis when it was in its very first steps.

I wish to thank my colleagues and friends from the office and the *tuperless* people for the long journey we have shared together during all these years.

I would like to express my gratitude to Toni Felguera and Dr. Mario Reyes from Barcelona Digital Technology Centre for giving me the opportunity of collaborating with them while finishing this thesis.

I want to thank DANTE, CESCA, UPCnet and Barcelona Digital Technology Centre for giving me access to their respective networks to collect the traffic traces that I have used to evaluate the work performed in this thesis.

I would also like to thank the European COST Action IC0703 and ETH Zürich University for giving me the opportunity of doing an internship in Cambridge (grants refs. 5202 and 5663) and Zurich, respectively.

This thesis was mainly funded by a UPC research grant as well as by the Spanish Ministry of Education (contract TEC2011-27474) and the Catalan Government

(contract 2009SGR-1140).

Finally, I want to express my most sincere gratitude to my family, in particular to my parents and sister for being always there. Also, I would like to especially thank Laura for dragging me into the research world and for helping me along the way and Leti for giving me the necessary strength to keep going until the end of this thesis. *Moltíssimes gràcies a tots!*

Abstract

Network monitoring has always been a topic of foremost importance for both network operators and researchers for multiple reasons ranging from anomaly detection to traffic classification or capacity planning. Nowadays, as networks become more and more complex, traffic increases and security threats reproduce, achieving a deeper understanding of what is happening in the network has become an essential necessity.

In particular, due to the considerable growth of cybercrime, research on the field of anomaly detection has drawn significant attention in recent years and tons of proposals have been made. All the same, when it comes to deploying solutions in real environments, some of them fail to meet some crucial requirements.

Taking this into account, this thesis focuses on filling this gap between the research and the non-research world. Prior to the start of this work, we identify several problems. First, there is a clear lack of detailed and updated information on the most common anomalies and their characteristics. Second, unawareness of sampled data is still common although the performance of anomaly detection algorithms is severely affected. Third, operators currently need to invest many work-hours to manually inspect and also classify detected anomalies to act accordingly and take the appropriate mitigation measures. This is further exacerbated due to the high number of false positives and false negatives and because anomaly detection systems are often perceived as extremely complex black boxes.

Analysing an issue is essential to fully comprehend the problem space and to be able to tackle it properly. Accordingly, the first block of this thesis seeks to obtain detailed and updated real-world information on the most frequent anomalies occurring in backbone networks. It first reports on the performance of different commercial systems for anomaly detection and analyses the types of network anomalies detected. Afterwards, it focuses on further investigating the characteristics of the anomalies found in a backbone network using one of the tools for more than half a year. Among other results, this block confirms the need of applying sampling in an operational environment as well as the unacceptably high number of false positives and false negatives still

reported by current commercial tools.

On the whole, the presence of sampling in large networks for monitoring purposes has become almost mandatory and, therefore, all anomaly detection algorithms that do not take that into account might report incorrect results. In the second block of this thesis, the dramatic impact of sampling on the performance of well-known anomaly detection techniques is analysed and confirmed. However, we show that the results change significantly depending on the sampling technique used and also on the common metric selected to perform the comparison. In particular, we show that, *Packet Sampling* outperforms *Flow Sampling* unlike previously reported. Furthermore, we observe that *Selective Sampling (SES)*, a sampling technique that focuses on small flows, obtains much better results than traditional sampling techniques for scan detection. Consequently, we propose *Online Selective Sampling*, a sampling technique that obtains the same good performance for scan detection than *SES* but works on a per-packet basis instead of keeping all flows in memory. We validate and evaluate our proposal and show that it can operate online and uses much less resources than *SES*.

Although the literature is plenty of techniques for detecting anomalous events, research on anomaly classification and extraction (e.g., to further investigate what happened or to share evidence with third parties involved) is rather marginal. This makes it harder for network operators to analise reported anomalies because they depend solely on their experience to do the job. Furthermore, this task is an extremely time-consuming and error-prone process. The third block of this thesis targets this issue and brings it together with the knowledge acquired in the previous blocks. In particular, it presents a system for automatic anomaly detection, extraction and classification with high accuracy and very low false positives. We deploy the system in an operational environment and show its usefulness in practice.

The fourth and last block of this thesis presents a generalisation of our system that focuses on analysing all the traffic, not only network anomalies. This new system seeks to further help network operators by summarising the most significant traffic patterns in their network. In particular, we generalise our system to deal with big network traffic data. In particular, it deals with src/dst IPs, src/dst ports, protocol, src/dst Autonomous Systems, layer 7 application and src/dst geolocation. We first deploy a prototype in the European backbone network of GÉANT and show that it can process large amounts of data quickly and build highly informative and compact reports that are very useful to help comprehending what is happening in the network. Second, we deploy it in a completely different scenario and show how it can also be successfully used in a real-world use case where we analyse the behaviour of highly distributed devices related with a critical infrastructure sector.

Resum

La monitorització de xarxa sempre ha estat un tema de gran importància per operadors de xarxa i investigadors per múltiples raons que van des de la detecció d'anomalies fins a la classificació d'aplicacions. Avui en dia, a mesura que les xarxes es tornen més i més complexes, augmenta el trànsit de dades i les amenaces de seguretat segueixen creixent, aconseguir una comprensió més profunda del que passa a la xarxa s'ha convertit en una necessitat essencial.

Concretament, degut al considerable increment del ciberactivisme, la investigació en el camp de la detecció d'anomalies ha crescut i en els darrers anys s'han fet moltes i diverses propostes. Tot i això, quan s'intenten desplegar aquestes solucions en entorns reals, algunes d'elles no compleixen alguns requisits fonamentals.

Tenint això en compte, aquesta tesi se centra a omplir aquest buit entre la recerca i el món real. Abans d'iniciar aquest treball es van identificar diversos problemes. En primer lloc, hi ha una clara manca d'informació detallada i actualitzada sobre les anomalies més comuns i les seves característiques. En segona instància, no tenir en compte la possibilitat de treballar amb només part de les dades (mostreig de trànsit) continua sent bastant estès tot i el sever efecte en el rendiment dels algorismes de detecció d'anomalies. En tercer lloc, els operadors de xarxa actualment han d'invertir moltes hores de feina per classificar i inspeccionar manualment les anomalies detectades per actuar en conseqüència i prendre les mesures apropiades de mitigació. Aquesta situació es veu agreujada per l'alt nombre de falsos positius i falsos negatius i perquè els sistemes de detecció d'anomalies són sovint percebuts com caixes negres extremadament complexes.

Analitzar un tema és essencial per comprendre plenament l'espai del problema i per poder-hi fer front de forma adequada. Per tant, el primer bloc d'aquesta tesi pretén proporcionar informació detallada i actualitzada del món real sobre les anomalies més freqüents en una xarxa troncal. Primer es comparen tres eines comercials per a la detecció d'anomalies i se n'estudien els seus punts forts i febles, així com els tipus d'anomalies de xarxa detectats. Posteriorment, s'investiguen les característiques de

les anomalies que es troben en la mateixa xarxa troncal utilitzant una de les eines durant més de mig any. Entre d'altres resultats, aquest bloc confirma la necessitat de l'aplicació de mostreig de trànsit en un entorn operacional, així com el nombre inacceptablement elevat de falsos positius i falsos negatius en eines comercials actuals.

En general, el mostreig de trànsit de dades de xarxa (és a dir, treballar només amb una part de les dades) en grans xarxes troncals s'ha convertit en gairebé obligatori i, per tant, tots els algorismes de detecció d'anomalies que no ho tenen en compte poden veure seriosament afectats els seus resultats. El segon bloc d'aquesta tesi analitza i confirma el dramàtic impacte de mostreig en el rendiment de tècniques de detecció d'anomalies plenament acceptades a l'estat de l'art. No obstant, es mostra que els resultats canvien significativament depenent de la tècnica de mostreig utilitzadai també en funció de la mètrica usada per a fer la comparativa. Contràriament als resultats reportats en estudis previs, es mostra que *Packet Sampling* supera *Flow Sampling*. A més, a més, s'observa que *Selective Sampling (SES)*, una tècnica de mostreig que se centra en mostrejar fluxes petits, obté resultats molt millors per a la detecció d'escanejos que no pas les tècniques tradicionals de mostreig. En conseqüència, proposem *Online Selective Sampling*, una tècnica de mostreig que obté el mateix bon rendiment per a la detecció d'escanejos que *SES*, però treballa paquet per paquet enlloc de mantenir tots els fluxes a memòria. Després de validar i evaluar la nostra proposta, demostrem que és capaç de treballar *online* i utilitza molts menys recursos que *SES*.

Tot i la gran quantitat de tècniques proposades a la literatura per a la detecció d'esdeveniments anòmals, la investigació per a la seva posterior classificació i extracció (p.ex., per investigar més a fons el que va passar o per compartir l'evidència amb tercers involucrats) és més aviat marginal. Això fa que sigui més difícil per als operadors de xarxa analalitzar les anomalies reportades, ja que depenen únicament de la seva experiència per fer la feina. A més a més, aquesta tasca és un procés extremadament lent i propens a errors. El tercer bloc d'aquesta tesi se centra en aquest tema tenint també en compte els coneixements adquirits en els blocs anteriors. Concretament, presentem un sistema per a la detecció extracció i classificació automàtica d'anomalies amb una alta precisió i molt pocs falsos positius. Adicionalment, despleguem el sistema en un entorn operatiu i demostrem la seva utilitat pràctica.

El quart i últim bloc d'aquesta tesi presenta una generalització del nostre sistema que se centra en l'anàlisi de tot el trànsit, no només en les anomalies. Aquest nou sistema pretén ajudar més als operadors ja que resumeix els patrons de trànsit més importants de la seva xarxa. En particular, es generalitza el sistema per fer front al "big data" (una gran quantitat de dades). En particular, el sistema tracta IPs origen i destí, ports origen i destí, protocol, Sistemes Autònoms origen i destí, aplicació que ha generat el trànsit i finalment, dades de geolocalització (també per origen i destí). Primer,

despleguem un prototip a la xarxa europea per a la recerca i la investigació (GÉANT) i demostrem que el sistema pot processar grans quantitats de dades ràpidament així com crear informes altament informatius i compactes que són de gran utilitat per ajudar a comprendre el que està succeint a la xarxa. En segon lloc, despleguem la nostra eina en un escenari completament diferent i mostrem com també pot ser utilitzat amb èxit en un cas d'ús en el món real en el qual s'analitza el comportament de dispositius altament distribuïts.

Resumen

La monitorización de red siempre ha sido un tema de gran importanciancia para operadores de red e investigadores para múltiples razones que van desde la detección de anomalías hasta a la clasificación de aplicaciones. Hoy en día, a medida que las redes se vuelven más y más complejas, aumenta el tráfico de datos y las amenazas de seguridad siguen creciendo, conseguir una comprensión más profunda de lo que ocurre en la red se ha convertido en una necesidad esencial.

Concretamente, debido al considerable incremento del ciberactivismo, la investigación en el campo de la detección de anomalías ha crecido y en los últimos años se han hecho muchas y varias propuestas. Aun así, cuando se intentan desarrollar estas soluciones en entornos reales, algunas de ellas no cumplen algunos requisitos fundamentales.

Teniendo esta problemática en cuenta, esta tesis se centra en llenar este vacío entre la investigación y el mundo real. Antes de iniciar este trabajo se identificaron varios problemas. En primer lugar, hay una clara falta de información detallada y actualizada sobre las anomalías más comunes y sus características. En segunda instancia, no tener en cuenta la posibilidad de trabajar con sólo parte de los datos (muestreo de tráfico) continúa siendo bastante extendido a pesar del severo efecto en el rendimiento de los algoritmos de detección de anomalías. En tercer lugar, los operadores de red actualmente deben invertir muchas horas de trabajo para clasificar e inspeccionar manualmente las anomalías detectadas para actuar en consecuencia y tomar las medidas apropiadas de mitigación. Esta situación se ve agravada por el alto número de falsos positivos y falsos negativos y porque los sistemas de detección de anomalías son a menudo percibidos como cajas negras extremadamente complejas.

Analizar un tema es esencial para comprender plenamente el espacio del problema y poder hacerle frente de forma adecuada. Por lo tanto, el primer bloque de esta tesis proporciona información detallada y actualizada del mundo real sobre las anomalías más frecuentes en una red troncal. Primero se comparan tres herramientas comerciales para la detección de anomalías y se estudian sus puntos fuertes y débiles, así como los

tipos de anomalías de red detectadas. Posteriormente, se investigan las características de las anomalías que se encuentran en la misma red troncal utilizando una de las herramientas durante más de medio año. Entre otros resultados, este bloque confirma la necesidad de la aplicación de muestreo de tráfico en un entorno operacional así como el número inaceptablemente elevado de falsos positivos y falsos negativos en herramientas comerciales actuales.

En general, el muestreo de tráfico de datos de red (es decir, trabajar sólo con una parte de los datos) en grandes redes troncales se ha convertido en casi obligatorio y, por lo tanto, todos los algoritmos de detección de anomalías que no lo tienen en cuenta pueden ver seriamente afectados sus resultados. El segundo bloque de esta tesis analiza y confirma el dramático impacto del muestreo en el rendimiento de técnicas de detección de anomalías plenamente aceptadas en el estado del arte. No obstante, se muestra que los resultados cambian significativamente dependiendo de la tácnica de muestreo utilitzada y también en función de la métrica usada para hacer la comparativa. Contrariamente los resultados en estudios previos, se muestra que *Packet Sampling* supera *Flow Sampling*. Además, se observa que *Selective Sampling (SES)*, una técnica de muestreo que se centra en muestrear flujos de datos pequeños, obtiene resultados mucho mejores para la detección de escaneos que las técnicas tradicionales de muestreo. En consecuencia, proponemos *Online Selective Sampling*, una técnica de muestreo que obtiene el mismo buen rendimiento para la detección de escaneos que *SES*, pero trabaja paquete por paquete en lugar de mantener todos los flujos en memoria. Después de validar y evaluar nuestra propuesta, demostramos que es capaz de trabajar *online* y utiliza muchos menos recursos que *SES*.

A pesar de la gran cantidad de técnicas propuestas en la literatura para la detección de eventos anómalos, la investigación para su posterior clasificación y extracción (p.ej., para investigar más a fondo lo que pasó o para compartir la evidenciancia de un ataque con tercerosinvolucrados) es más bien marginal. Esto hace que sea más difícil para los operadores de red analalitzar las anomalías reportadas, ya que dependen únicamente de su experiencia para hacer el trabajo. Además, esta tarea es un proceso extremadamente lento y propenso a errores. El tercer bloque de esta tesis se centra en este tema teniendo también cuenta los conocimientos adquiridos en los bloques anteriores. Concretamente, presentamos un sistema para la detección, extracción, y clasificación automática de anomalías con una alta precisión y muy pocos falsos positivos. Adicionalmente, desplegamos el sistema en un entorno operativo y demostramos su utilidad práctica.

El cuarto y último bloque de esta tesis presenta una generalización de nuestro sistema que se centra en el análisis de todo el tráfico, no sólo en las anomalías. Este nuevo sistema pretende ayudar m a los operadores ya que resume los patrones de tráfico

más importantes de su red. En particular, se generaliza el sistema para hacer frente al "big data" "(una gran cantidad de datos). En particular, el sistema trata IPs origen y destino, puertos origen y destino, protocolo, Sistemas Autónomos origen y destino, aplicacinón que ha generado el tráfico y finalmente, datos de geolocalitzación (también para el origen y el destino). Primero, desplegamos un prototipo en la red europea para la investigación y el desarrollo (GÉANT) y demostramos que el sistema puede procesar grandes cantidades de datos rapidamente así como crear informes altamente informativos y compactos que son de gran utilidad para ayudar a comprender lo que está sucediendo en la red. En segundo lugar, desplegamos nuestra herramienta en un escenario completamente diferente y mostramos cómo también puede ser utilizado con éxito en un caso del mundo real en el que se analiza el comportamiento de dispositivos altamente distribuidos.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AC Anomaly Classification

AD Anomaly Detection

AE Anomaly Extraction

ADSL Asymetric Digital Subscriber Line

CCABA Advanced Broadband Communications Center

CESCA Supercomputing Center of Catalonia

CPU Central Processing Unit

DANTE Deivery of Advanced Technology to Europe (company)

DoS Denial of Service Attack

DDoS Distributed Denial of Service Attack

DNS Domain Name Server

FaRNet Fast Recognition of High Multi-dimensional Network Traffic Patterns

FIM Frequent Item-set Mining

FI Frequent Item-set

FN False Negatives

FP False Positives

FS Flow Sampling

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IANA Internet Assigned Numbers Authority

IDS Intrusion Detection System

ML Machine Learning

NR NetReflex

NREN National Research and Education Network

NS Network Scan Attack

OSES Online Selective Sampling

P2P Peer-to-Peer

PF PeakFlow SP

PoP Point of Presence

PS Packet Sampling / Port Scan Attack

PCA Principal Component Analysis

SES Selective Sampling

SFF Same Fraction of Flows

SFP Same Fraction of Packets

SMS Smart Sampling

SNMP Simple Network Management Protocol

SSH Secure Shell

RPC Remote Procedure Call

SMB Server Message Block

SQL Structured Query Language

SW StealthWatch

TAPS Time-based Access Pattern Sequential Hypothesis Testing

TCP Transmission Control Protocol

TP True Positives

TRW Threshold Random Walk

UDP User Datagram Protocol

UPC Universitat Politècnica de Catalunya BarcelonaTech

Chapter 1

Introduction

In recent years, understanding what is really happening in a network is increasingly becoming more and more complex due to the ever-growing list of applications shaping today's Internet traffic. Consequently, traffic monitoring and analysis has become crucial for tasks ranging from intrusion detection to traffic engineering, capacity planning or traffic classification. In particular, due to the tremendous rise of cyber attacks worldwide [1], the amount of research focused on anomaly detection has been vast. However, lots of proposals often do not comply with some real world requirements when deployed in large backbone networks. In this thesis, we analyse and address some of these challenging issues.

This chapter discusses the motivations and challenges behind this dissertation and also provides an overview of the thesis and its main contributions. Finally, it explains how the present manuscript is structured.

1.1 Motivations

Network operators have always been interested in keeping track of the anomalies happening in their network. Traditionally, they have focused on operational (e.g., link faults), or traffic and routing anomalies, observable via SNMP. More recently, there has been a business driver for observing anomalies related to security issues, network abuse, or IPR violation. The reason for investing in security, even in core networks, is that offering a more secure network (i.e., protecting customers from external or internal threats) is becoming a differentiating factor for ISPs, already offering managed security services to their business customers. Furthermore, commercial peering agreements between ISPs often include commitments to avoid transferring potentially

harming traffic (e.g., DoS attacks).

Although it is plenty of solutions for anomaly detection, some of them are often based on outdated assumptions or unrealistic requirements. This is because detailed information about the types and characteristics of current network anomalies happening in large networks is rather rare in the literature [2, 3, 4, 5, 6]. As a result, our understanding of these anomalies happening in backbone networks is often incomplete or simply outdated. Taking into account the incredible complexity of the Internet today and how fast new threats and applications emerge daily, being aware of current trends is crucial. In this thesis, we address this by performing an in-depth analysis on the anomalies found in a large backbone network for more than half a year.

Many anomaly detection proposals have not been yet widely adopted by network operators because of several practical problems regarding its practical applicability in operational environments. Namely 1) they generate a large number of false positives, and false negatives, 2) they do not take traffic sampling into account, 3) they are often seen as extremely complex mechanisms that act as black boxes, 4) they do not distinguish different anomaly types (most systems differentiate between normal and anomalous traffic), and 5) they do not provide any mechanisms for further analysing an anomaly once it has been detected, among others.

Due to the almost mandatory usage of sampling in high-speed links, anomaly detection has become an even more challenging task. Robustness against traffic sampling is now a topic of paramount importance since network operators tend to apply aggressive sampling rates when using monitoring tools like NetFlow [7] (e.g., take 1 packet out of 1000) in order to handle worst case scenarios and network attacks. For this reason, it is fundamental to build sampling-resilient anomaly detection mechanisms.

In particular, in this thesis we are interested in analysing the performance of scan detection algorithms under sampling. We focus on network scanning for several reasons. Firstly, they are frequently the prequel of other attacks (e.g., worm propagation) and, therefore, there is general interest in detecting them reliably. Secondly, scanning activities represent more than 80% of the anomalies on the Internet according to a recent study [8]. Moreover, scans can put monitoring platforms in serious trouble (the nature of this sort of anomalies can easily overflow flow tables due to the potentially large set of new flows generated by the scanner). Several methods for scan detection have been proposed in the literature. The straightforward approach flags a scanner when it connects to more than a certain number of destinations during a fixed interval of time. This method is implemented in the well-known Snort IDS [9]. The mechanisms analysed in this thesis (TRW [10] and TAPS [11]) are more complex and have shown to be more effective. In particular, TRW is implemented in the also well-known Bro IDS [12]. The idea of TRW is that a scanner will fail more connections than a benign

host, thus classifying a host as non-legitimate when it makes too many consecutive failed connections. TAPS is based on the observation that scanners visit many more destination IPs vs. ports than normal hosts (or the reverse, depending on the type of scan; vertical or horizontal). Likewise TRW, when this condition is accomplished several times, TAPS will report the host generating such flows as a scanner.

Several studies have analysed the impact of sampling on anomaly detection [13, 14, 15, 16, 17, 18] and few have focused specifically on scan detection [14, 13]. In particular, the impact of *Packet Sampling* on TRW and TAPS was analysed in [14]. It was observed that both the false positives and the false negatives increased significantly with sampling for TRW. It was also concluded that TAPS was more resilient to sampling than TRW because even though TRW had higher success ratio, TAPS exhibited a significantly lower ratio of false positives. The same authors extended the analysis to several sampling mechanisms in [13]. It was shown that *Packet Sampling* introduced an important bias in the flow size distribution and that techniques targeting large flows were not convenient for scan detection (scanning attacks use small flows). Consequently, it was concluded that *Flow Sampling* was the best choice for scan detection under sampling.

While previous works (e.g., [13]) used only the same percentage of sampled flows as the common metric to compare the different sampling methods, in this thesis we also use the same fraction of packets. Since every packet must be processed by the router, it is also important to compare the accuracy of all sampling methods according to the ratio of sampled packets. The motivation of this came from the fact that given a flow sampling rate, the fraction of analysed packets is significantly different among the sampling methods, which results in an unfair comparison, specially for packet-based sampling techniques like *Packet Sampling*. For instance, according to our traces, sampling 10% of the flows results only in 3.01% of sampled packets for *Packet Sampling*, while *Flow Sampling* gets 9.2% and other sampling techniques sample an even larger percentage. Unlike previously reported [14, 13], after this analysis we were able to observe that *Packet Sampling* could perform better than *Flow Sampling* under certain scenarios.

Selective Sampling [18] is a recently proposed sampling technique that targets small flows, which are normally used for launching scanning attacks. However, this technique first needs to capture all packets and then it samples entire flows. The main issue of this scheme is that all the flows must be kept in main memory and it is not implementable in the packet-based NetFlow deployed in most routers. In this thesis, we propose a sampling technique that has the same goal (sample small flows) but does not need to capture all the traffic. By taking per-packet decisions, our method requires significantly less resources while keeping the same good performance for scan

detection showed by *Selective Sampling* and being implementable in NetFlow.

In addition to anomaly detection, anomaly classification, i.e., automatically identifying the type of a detected anomaly (e.g., scan attack), has been marginally studied in the past. Lakhina et al. [19] cluster the output of a PCA-based anomaly detector to identify anomalies with similar behaviour. However, human intervention is necessary to find out the correspondence between each reported cluster and the high-level anomaly that it is describing. Tellenbach et al. [20] classify changes to generalized entropy metrics of traffic feature distributions to identify the type of detected anomalies. They demonstrate that this approach can classify with accuracy of around 85% synthetic anomalies. Choi et al. [21] make use of parallel coordinate plots to find unique patterns of attacks that are easy to recognize visually by a human expert. In contrast, in this thesis we are able to automatically classify network anomalies without requiring later manual inspection.

Moreover, analysing the root cause of network anomalies is a matter of increasing importance. Once an anomaly has been flagged, network operators need to diagnose the type of the anomaly in order to act accordingly and take the appropriate mitigation measures. Presently, operators typically need to invest many work-hours to manually inspect and classify detected anomalies, which is an extremely complex, slow, and expensive process [22]. The problem of anomaly extraction has been recently treated in the literature [23, 24]. Among the existing proposals, the most relevant to our work [23] uses a frequent item-set mining (FIM) algorithm to identify the flows related to an anomaly from a given hint (e.g., an involved IP address) provided by an external anomaly detector. In this thesis, we present a FIM-based system that is not only able to extract these flows related to an anomaly for further investigation but also performs anomaly detection and classification by combining data mining with machine learning.

After deploying such tool in an operational environment and confirming its usefulness for anomaly detection, extraction and classification, we realized that the used techniques were powerful enough to be used for a broader and more challenging goal not limited only to network anomalies. In particular, FIM-based algorithms have the ability of naturally handling larger amounts of data, which makes them specially suitable for processing network traffic. In recent years, the Internet traffic has increased and its mix has changed significantly. In such a rapidly changing environment with so much data, it is critical to build traffic profiling tools that are able to process big network traffic data efficiently to extract knowledge about what is happening in a network. Cyber-activism, social networks, peer-to-peer applications or streaming services are only a few examples of the ever-growing list of applications, services and network attacks that mold Internet traffic today. Furthermore, existing applications and attacks continuously change their behaviour while new applications, services and cyber-threats

emerge every day.

The most basic traffic profiling technique (and likely the most widely-used one) is extracting heavy hitter reports about the top, e.g., IP addresses, that drive network utilization. However, a simple traditional report of the, e.g., top source IP addresses, does not give any information about what these hosts are actually doing, which reduces a lot the utility of such reports. To address this problem, previous work has studied the problem of finding *multi-dimensional* and *hierarchical* heavy hitters (HHHs). Notably, AutoFocus [25] is the best known tool for finding multi-dimensional HHHs. An HHH is an aggregate (e.g., an IP address prefix) on a hierarchy that appears frequently under certain constrains. The input data can be single- or multi-dimensional (e.g., IP address pairs) which gives rise to single- and multi-dimensional HHHs.

Although AutoFocus is a state-of-the-art HHH-based traffic profiling tool, it has some important limitations. First, its computational overhead grows exponentially with the number of dimensions. Because of this, it is restricted to 5-dimensional HHHs (where the five dimensions correspond to the source and destination IP addresses, the port numbers and the protocol) and it is very hard to extend it with additional traffic features. In this thesis, we propose an extension of our system that is able to build reports that contain other relevant information, such as the layer 7 traffic application, the source and the destination Autonomous Systems (ASes) and the geographical location. This extension provides precise and compact traffic summaries of what is really happening in a network. For example, a report looks like "Univ.A Google Location-X Video", i.e., University A shares a significant amount of video traffic with a Google data center at Location-X. Second, even with 5-dimensional HHHs AutoFocus exhibits very high computational overhead and cannot typically meet *near real-time*[1] guarantees even at low input rates. The last part of this thesis presents this extension of our system and shows examples of real reports that are very useful for network operators.

1.2 Thesis Overview and Contributions

This thesis is about filling the gap between the real-world requirements from the network security industry and the research carried out in the anomaly detection area. The main hypothesis behind this dissertation is that practitioners and researchers working on anomaly detection do not go hand in hand due to a mismatch of such requirements. For example, the presence of sampling is often omitted in research proposals, which makes it difficult to deploy them in real environments, where their performance might

[1]We define *near real-time* as the requirement of fully processing an x-minute interval of traffic data in no longer than x minutes, where x is typically a small constant, like a 5-minute window.

be affected due to the aggressive sampling rates that are commonly used. Additionally, other factors such as the amount of false positives or the lack of support for assisting the still manual investigation of network anomalies pose an important breach between industry and research.

To fill this gap, the first part of the thesis studies 1) the real-world requirements from the point of view of a network operator and the performance of current commercial tools for anomaly detection and 2) the actual characteristics of anomalous traffic in the wild. Among others, we found that the amount of false positives and negatives is surprisingly high even in commercial tools. This shows current limitations that need to be addressed by researchers and also demonstrates that there is still a lot of work left in this area as latest research advances have not been fully applied in the industry. After identifying the mismatch of requirements and the characteristics of current anomalies, the second part of the thesis analyses state-of-the-art methods for anomaly detection under these real-world constraints. We found that, unlike previously reported, Packet Sampling can be a good option for scan detection and also discovered that a data mining algorithm called frequent item-set mining, which had not been previously used in this area, is very convenient for anomaly detection and is also resilient to traffic sampling. All in all, we believe that having a clear understanding of the real requirements from the industry and knowing the characteristics of current anomalous traffic in the wild will help to guide research on anomaly detection towards more practical and actual problems faced by network operators and other practitioners in this area. Chapters 4, 5 and 6 of this thesis are a first step towards this direction. Next, we give an overview of each chapter and summarise the main contributions of this thesis.

The main goal of the first block (Chapter 3) is to provide a long-term study of current security-related anomalies happening in backbone networks. We analyse their types and how frequent they are. Additionally, we investigate the behaviour of each sort of attack and the most common origins and destinations of such anomalies, among others. Moreover, we report on the practical limitations of current commercial tools for anomaly detection in an operational environment, which is an important aspect to take into account and helps identifying areas where research still needs further improvement.

This block makes two contributions. First of all, although there has been considerable work in the design and evaluation of anomaly detection methods, information regarding the type and characteristics of network anomalies in operational networks does not exist in the literature. To the best of our knowledge, this is the first work that provides this sort of feedback to the research community. For example, we show that scans are the most frequent attacks in the European backbone network of GÉANT. This type of knowledge is essential in order to guide research towards particularly prob-

lematic anomalies in the real world (e.g., improving the detection of scanning attacks in this case). Additionally, other information that we report such as the anomaly distribution and the characteristics of each type of anomaly is also unknown and can be of great interest for any researcher working on anomaly detection in order to e.g., design better detection algorithms.

Second, researchers usually do not have access to data from backbone networks or the possibility of testing commercial tools (they are, in general, too expensive). As a consequence, current limitations of commercial solutions are unknown by the community. Thus, the information provided in this block will be a precious resource for any researcher working on anomaly detection. Providing this feedback can probably help directing further investigations towards solving specific real world problems. For example, we found that the overlap among the anomalies detected by current commercial tools is surprisingly low, which indicates a significantly high number of false negatives. Moreover, we show that the overall number of false positives is still important and report on the false alarms per anomaly type. This information highlights that some areas require further research by the anomaly detection community to overcome the limitations of current anomaly detection mechanisms deployed in operational environments.

The second block (Chapter 4) is devoted to analyse the impact of sampling on already proposed algorithms for network anomaly detection. First, we analyse the effect of different traffic sampling techniques on well-known scan detection mechanisms and conclude that *Packet Sampling* performs better than *Flow Sampling* under the same percentage of packets (previous studies used the same percentage of flows). Second, we propose a new sampling technique based on *Selective Sampling* and show that it uses far less resources while keeping the same good accuracy and being implementable in routers.

Previous works had reported instead that *Flow Sampling (FS)* was preferable over *Packet Sampling (PS)* for scan detection [14, 13]. In particular, it had been concluded that *PS* offered a very poor performance for scan detection [14, 13]. This was a major concern for network security as routers only implement packet-based sampling. However, in this block we reach different conclusions by using a different methodology. Specifically, our first contribution on this topic is to show that *PS* performs better than *FS* under the same percentage of packets. Previous studies used the same percentage of flows for the comparison. Overall, we show that *PS* is neither better nor worse than *FS* as they both offer similar performances in terms of accuracy. Nonetheless *PS* has two key advantages over *FS*: it is already supported by routers and its computational complexity is lower. This knowledge is an important contribution not only for the research community but also for router vendors, because based on previous studies, the

bottom line was that routers should be upgraded to support *FS* in order to increase the performance of scan detection algorithms. We show instead that *PS* performs similarly to *FS* for scan detection, which indicates that this update would be not only arguable but would consume much more resources in the router.

Out of the four sampling techniques we analysed, *Selective Sampling (SES)* [17] offered the best performance for scan detection. However, this technique keeps all the traffic in memory as it needs to know the final size of the flow, which is very costly in terms of memory. Moreover, *SES* is not implementable in NetFlow as it works on a per-flow basis. Our second contribution of this block is a sampling technique that keeps the same accuracy that *SES* but does not require to store all the traffic, which results in several benefits in terms of resource consumption. We propose *Online Selective Sampling*, a packet-based sampling technique based on *SES* that 1) is able to obtain the same good results for scan detection, 2) is faster and uses less memory and 3) works online and can be implemented in the packet-based Sampled NetFlow [26], which is deployed in most routers.

The third block (Chapter 5) is centered on building a system for automatic anomaly detection, extraction and classification that is easily comprehensible for network operators. The system proposed uses data mining for anomaly detection and extraction, and machine learning for anomaly classification. We show that our proposal has high accuracy and very low false positives by deploying it in the Catalan NREN.

Machine learning (ML) mechanisms have been largely used for anomaly detection. The key novelty and main contribution of this chapter resides on using ML together with a technique called frequent item-set mining (FIM). To the best of our knowledge, this is the first work that uses FIM for anomaly detection. The main advantage of FIM is that it is able to characterize anomalies very well. This is due to the very nature of FIM, which finds strong and frequent correlations among elements. Anomalies are composed by sets of flows sharing certain features (e.g., the destination IP and port for a Distributed Denial-of-Service or the source and the destination IPs for a vertical scan). FIM accurately extracts all these flows generating an attack and therefore it is very convenient to describe it (e.g., average flow size of the attack traffic or number of flows that contain the attack). These sets of flows, called frequent item-sets, together with their description, allow ML to characterize different anomaly types very accurately. For example a frequent item-set summarising more than e.g., 3000 flows from a fixed source IP to a particular destination IP, using the same source port, but targeting different destination ports and using an average packet size of 29 bytes (UDP with 1 byte of payload) is a clear indicator of malicious activity (a vertical scan in this particular case). In contrast, using different types of aggregations (e.g., by source or destination IP) would be less accurate as aggregations are pre-defined and, therefore,

they can contain few or too much traffic, which, in turn, might include a mixture of both legitimate and anomalous flows.

Finally, the fourth block (Chapter 6) shows how the methodology presented in the previous block can be generalised to build a system (*FaRNet*) that deals with all the traffic, not only network anomalies. In particular, we show that FIM can easily handle the huge amount of traffic in backbone networks (e.g., GÉANT) and produce highly synthetic reports that are able to summarise what is happening in the network very compactly. We show that the flexibility and the scalability of our system, which is not limited to the traditional 5-tuple like AutoFocus, makes it possible to investigate many other interesting features (e.g., geolocation or traffic application) and build much more informative and useful reports that can help network operators to better comprehend what type of traffic is shaping their networks. We successfully deploy a prototype of *FaRNet* in the European backbone network of GÉANT. Furthermore, we also present another deployment of the same tool in a different environment related with monitoring the behaviour of highly distributed devices in a critical infrastructure sector (see Appendix A).

Our main contribution in this last block is to show how FIM is able to deal with much more data than proposed algorithms for reporting exact hierarchical heavy hitters (HHH) [27, 28, 25]. We show how our FIM-based proposed algorithm for dealing with hierarchical data is far superior to the well-known AutoFocus [25, 29], which is the state-of-the-art HHH-based tool. While AutoFocus works offline and already presents heavy computational overhead for analysing 5 dimensions, we show that FIM scales well as it is able to process up to 10 dimensions in near real-time.

Additionally to the better performance offered by our FIM-based proposal, mining five new dimensions (source and destination ASes, source and destination geolocation and layer 7 application on top of the 5-tuple), provides much more synthetic, relevant and useful traffic summaries with respect to AutoFocus reports. For example, we can find patterns such as e.g., *Spain is exchanging an important amount of P2P traffic with the United States using a specific source AS*. In contrast, AutoFocus would only report activity at a very low level of granularity, i.e., interaction among certain IPs, which, in turn, would be split into multiple lines of the summary. This report would not provide an extremely valuable knowledge to a network operator. Furthermore, while AutoFocus does not scale to more dimensions, our FIM-based *FaRNet* could be easily extended to deal with other hierarchical elements. For instance, it could be used to correlate network traffic (e.g., NetFlow) with routing data (e.g., AS-paths), which could be utilized to perform an analysis on the most heavily used paths (useful for traffic engineering or cost reduction).

1.3 Thesis Outline

This thesis dissertation is organised as follows. First, Chapter 2 gives the necessary background on the algorithms analysed in this thesis and describes the scenarios used throughout this manuscript. Next, the four core blocks of the thesis are covered. First, Chapter 3 presents the Anomaly Analysis block, where a long-term analysis on the network anomalies found in the European-wide backbone network of GÉANT is reported. This is based on the work published in [22, 8]. Next, Chapter 4 first analyses the impact of several sampling techniques on two well-known scan detection algorithms. Afterwards, based on the results, the second part of the block describes our new packet-based sampling technique proposal, which works remarkably well for scan detection under sampling. This part of the thesis is published in [30, 31]. Chapter 5 presents our system for anomaly detection, extraction and classification based on frequent item-set mining and machine learning. Its design, evaluation and deployment have been published in [32, 33, 34]. Next, Chapter 6 explains how the system presented in the previous chapter can be successfully generalised to analyse big network traffic data and produce highly compact traffic summaries, which are very helpful for network operators to understand what is happening in their networks. Part of this work was published in [35] and the full version was submitted to a journal and is currently under review (see Appendix B). Finally, Appendix A presents a real world use case that shows how the methodology presented in chapters 5 and 6 can be successfully used in a completely different environment to analyse the behaviour of highly distributed devices related with a critical infrastructure sector. Ultimately, Chapter 6.5 covers the related work and Chapter 7 concludes this thesis and presents some ideas for future work.

Chapter 2

Background

This chapter first describes the different types of network anomalies (Section 2.1), sampling techniques (Section 2.2), scan detection mechanisms (Section 2.3) and frequent item-set mining algorithms (Section 2.4) analysed in this thesis. Finally, Section 2.5 explains the scenarios and datasets used for the experiments of this thesis.

2.1 Network Anomalies

We focus on some of the most common types of network attacks: *Network Scan*, *Port Scan*, *Denial-of-Service* and *Distributed Denial-of-Service*. Note that all these anomalies are malicious attacks. Other anomalies such as *Flash Crowds*, which are not ordinary but do not have a malicious purpose, are not within the scope of this thesis. Next, we describe each anomaly type.

2.1.1 Network Scans

NS or horizontal scans are carried out by sending probes to identify running services on a network. Those probes are always sent to one specific port (or a few), but to a multitude of destinations. The most common are *SYN scans* and *ACK scans* but other (not necessarily protocol legitimate) TCP flags combinations can be used.

2.1.2 Port Scans

PS or vertical scans are aimed at detecting running services on a specific machine. Essentially, this type of scan consists of sending a message to a huge amount of ports,

one at a time. The kind of response received indicates whether the port is used and can therefore be further probed for weakness. Frequently, these scans are carried out via UDP with 1 byte of payload packets (total packet size of 29 bytes).

2.1.3 Denial-of-Service

A *DoS* is an attack on a computer system that floods the network or the end system. They are attempts to make a resource unavailable to its users. Most common sub-categories are *DoS UDP Floods* and *DoS TCP SYN Floods*. A *UDP Flood* attack can be carried out with a large number of packets with a small byte size or with packets with a large byte size. The later type may look similar to a legitimate" heavy UDP transfer (e.g., bandwidth tests, high-volume P2P transfers, streaming applications or complex data transfers. A *TCP SYN Flood* attack is carried out by sending a succession of SYN requests to a target system. Some methods (e.g., usage of syn cookies, tuning of TCP parameters, use of stateful firewalls) can help to reduce the effects of this attack. However, this attack is still largely used.

2.1.4 Distributed DoS

DDoS attacks can take the forms described for DoS, but the senders are a multitude of (often compromised) systems attacking a single target. The effects of distributed attacks are nastier and their mitigation more difficult.

2.1.5 Others

This category was used to separate traffic that looked suspicious but did not fit in the groups above.

2.2 Sampling Techniques

In this section, we describe the following five sampling techniques: *Sampled NetFlow*, *Packet Sampling* (*PS*), *Flow Sampling* (*FS*), *Smart Sampling* (*SMS*) and *Selective Sampling* (*SES*). While the first four methods are well-known in the literature and commonly used, the fifth mechanism was also considered because it targets small flows, which are precisely the type of flows used by scanning attacks.

PS is widely used because of its low CPU consumption and memory requirements. Flow-based approaches (e.g., *SMS*) overcome some of the shortcomings of *PS* (e.g.,

they keep the original flow size distribution) but, in exchange, they have higher resource requirements. Thus, some trade-off between accuracy and resource requirements is needed. Next, we describe each sampling technique.

2.2.1 Sampled NetFlow

NetFlow [7] is a network protocol developed by *Cisco Systems* [36] used to collect IP traffic information on routers. It essentially aggregates incoming packets into flows (NetFlow records) and expires them every certain interval of time. The flow definition is not exactly the same among the different NetFlow versions but it is mainly composed by the source and destination IPs, the source and destination ports and the protocol. As we can deduce from the definition, a flow is considered to be unidirectional since a response flow would have the source and the destination IPs and ports reversed and the router would find no match with any previous flow, thus creating a new one. In order to avoid resource exhaustion on routers due to network anomalies or bursts in the traffic NetFlow allows sampling. When *Sampled NetFlow* [26] is activated, the router samples one packet out of N for every time interval configured.

2.2.2 Packet Sampling

Packet Sampling (PS) takes each packet with probability $0 \leq p < 1$. There are several versions [37] of this mechanism such as *Systematic Packet Sampling* and *Random Packet Sampling*. The first one samples packets in a deterministic way (take every k-th packet), while the second one depends on random decisions. Our *PS* implementation corresponds to the latter.

2.2.3 Flow Sampling

Flow Sampling (FS) takes each flow with probability $0 \leq p < 1$. In order to implement it efficiently, we used a hash-based technique called *Flowwise Packet Sampling* [38]. This technique does not need to collect all packets of each flow before sampling, i.e., it takes per-packet decisions.

2.2.4 Smart Sampling

Smart Sampling [39] (*SMS*) is a flow-based sampling technique that focuses on large flows and drops the small ones. In order to define what large means, a threshold z must be provided. z indicates the minimum size of the flows that will be sampled.

Flows with lower size will be discarded with a certain probability $p(x)$, which is inversely proportional to their size (x):

$$p(x) = \begin{cases} x/z & x < z \\ 1 & x \geq z \end{cases}$$

2.2.5 Selective Sampling

Selective Sampling [17] (*SES*) is also a flow-based sampling method. *SES* focuses on small flows, which are normally used for scanning. It uses three different parameters: z, c and n. The first one corresponds to the threshold that defines the size of a small flow (in packets). c is the probability of sampling a small flow (a flow with at most z packets). Finally, n is used to further regulate the percentage of non-small flows taken. The higher n, the lower the probability of sampling a flow with more than z packets. A flow of size x packets is sampled according to the following expression:

$$p(x) = \begin{cases} c & x \leq z \\ z/(n \cdot x) & x > z \end{cases}$$

2.3 Scan Detection mechanisms

Simple scan detection algorithms, like the one used by the Snort IDS, are useless nowadays since attackers can easily evade detection by reducing their scanning rate. There are other techniques capable of achieving higher rates of detection such as TRW and TAPS, which we analyse in this thesis. Next, we describe how these two scan detection algorithms work.

2.3.1 TRW

The main idea behind Threshold Random Walk (TRW) [10] is that one scanner will fail more connections than a legitimate client when trying to establish a connection. The method works under two hypothesis: the host is either legitimate (H_0) or malicious (H_1). Y_i is the final state of the i-th connection performed by a host:

$$Y_i = \begin{cases} 0 & \text{if connection successful} \\ 1 & \text{if connection failed} \end{cases}$$

$$Pr[Y_i = 0|H_0] = \theta_0 \qquad Pr[Y_i = 1|H_0] = 1 - \theta_0$$
$$Pr[Y_i = 0|H_1] = \theta_1 \qquad Pr[Y_i = 1|H_1] = 1 - \theta_1$$

Due to the fact that scanners will fail more connections than benign hosts, the following condition must be accomplished:

$$\theta_0 > \theta_1$$

Since it is possible to fail some connections even being a benign host, the decision of flagging a host as a scanner is not taken just after the first failure. For each source there is an accumulated ratio that is updated each time a flow ends.

$$\bigwedge(Y) = \prod_{i=1}^{n} \frac{Pr[Y_i|H_1]}{Pr[Y_i|H_0]}$$

The update is performed depending on the flow state: connection established or failed attempt. If a source IP keeps scanning, it will accumulate failed connections and, eventually, it will exceed the established threshold, thus being recognised as a scanner. Similarly, a legitimate host making several successful connections will be classified as non-scanner.

2.3.2 TAPS

Time-based Access Pattern Sequential hypothesis testing (TAPS) [11] is based on the observation that the ratio between the number of destination IPs and the number of destination ports (or the reverse) when the source IP is an scanner is significantly higher than the same ratio when the host is benign.

As the name suggest, TAPS is based on the same statistical method that TRW uses. It only changes two things: the way it detects a suspicious behaviour and the update mechanism. It checks the following fractions every t seconds (which is a configurable parameter given to the algorithm):

$$\frac{\#accessed\_IPs}{\#accessed\_ports} > k \qquad \frac{\#accessed\_ports}{\#accessed\_IPs} > k$$

The left side expression indicates a horizontal scan or a network scan, while right side expression means that a port scan or a vertical scan is going on (refer to Section 2.1 for details on types of scans).

When any of these two fractions is higher than a pre-configured threshold k, the per-source IP ratio is updated. Likewise TRW, when this accumulated value reaches a certain limit, that source is classified either as a scanner (upper threshold) or a legitimate host (lower threshold).

2.4 Frequent Item-Set Mining

The field of frequent item-set mining (FIM), comes from the data mining world and essentially tries to extract information from the data by finding out the most frequent relationships among different items. The original reason why this concept was created was to analyse the so called supermarket transaction data. Its purpose was to discover if there were any particular items bought frequently together by customers (e.g., chips and beer are bought together in 90% of the cases). This type of information can be extremely useful to, later on, e.g., offer promotions.

An item is considered frequent if it appears often in the data. "Often can be defined either as an absolute number (number of occurrences) or as a relative parameter (percentage relative to the overall amount of data) and it is called *minimum support (s)*. An item will be frequent if it has at least s occurrences or if it represents at least $s\%$ of the overall transactions. In terms of supermarket transaction data, a transaction would include all the items that a customer bought (one transaction per buyer). In summary, the problem of frequent item-set mining consists of looking for items occurring together (item-sets) above s.

The *downward-closure* property of FIM states that an item-set is frequent *iff* all its subsets are frequent. For this reason, all the subsets of a frequent item-set are also frequent and the output of FIM can include many redundant item-sets. To address this problem, there are three types of outputs that a frequent item-set miner can extract: all, closed or maximal [40]. The first group simply includes all item-sets while the last one extracts only the longest patterns in case of redundant sub-patterns. Closed item-sets try to find the optimal trade off between the excessive redundancy of extracting all item-sets and the inevitable loss of granularity given by maximal item-sets. Closed item-sets miners return the longest item-sets and, in addition, present also these patterns inside it that have higher frequency. For instance, suppose item $\{a\}$ has frequency 100 and that item-set $\{a,b\}$ has frequency 80. If $min\_sup = 75$, all frequent item-sets would be both $\{a\}$ and $\{a,b\}$. The maximal item-set miner would only show item-set $\{a,b\}$. The closed miner would return as well both item-sets because $\{a\}$ has higher frequency than its parent in terms of hierarchy ($\{a,b\}$). If, item-set $\{a\}$ had frequency 80 instead of 100, the closed miner would not show it since it would not add further information with respect to $\{a,b\}$ (in case of having the same frequency, show the longest item-set).

FIM mines efficiently an input set of transactions to discover frequent patterns. Each input transaction T consists of a set of l items $T = \{e_1, ..., e_l\}$. When using FIM for network traffic, we model each traffic flow as a transaction where the items correspond to different flow features. For example, an input transaction can consist of

the 5-tuple, i.e., , the source and destination IP addresses, the source and destination port numbers and the protocol.

A large number of FIM algorithms have been studied in the literature. In this thesis, we analyse Apriori [41], FP-growth [42], and Eclat [43] because they are the reference algorithms of the three main paradigms for computing a FIM solution. In addition, we also select RElim [44] and SaM [45], which are two highly-optimized variants of FP-growth. A survey of existing algorithms can be found in [46]. Next, we summarise the key features of each algorithm.

Apriori [41] is the first and simplest FIM algorithm. It works in breadth first order by iteratively merging frequent item-sets of increasing length. It starts by computing frequent item-sets of length one. Based on the *downward-closure* property, it then joins them to compute candidate item-sets of length two. Afterwards, it makes a pass over the input transactions and discards candidate item-sets that are not frequent. This procedure is repeated recursively until no more candidate item-sets can be generated. Apriori has two main drawbacks. First, it needs k passes over the input data, where k is the length of the longest item-set. Secondly, candidate generation and testing is extremely slow as the number of candidate item-sets can be very large. In order to overcome these issues, faster and more efficient algorithms have been proposed.

Eclat [43], instead of working with the typical horizontal representation of transactions, i.e., a list of items for each transaction, it uses a vertical layout, i.e., each item has an associated list of transaction identifiers where it appears in. It traverses the data in a depth first order and intersects the lists of transaction identifiers of the corresponding items for the counting.

FP-growth [42] uses a structure called Frequent Pattern Tree (FP-tree). In an FP-tree, those transactions sharing items will also share the same branch in the tree, which allows storing the data with higher compactness (especially for dense datasets). FP-growth skips the process of checking all candidate item-sets against all the database for each iteration, which is an extremely slow process and becomes untreatable as input data grows up or the *minimum support* s gets lower. FP-growth improves this by significantly reducing the possible candidates to that part of the database related to a particular item (called the *conditional pattern base*). Another factor that explains why FP-growth is faster than Apriori is that it only reads the database twice.

The main strength of RElim [44] lies on its simplicity. It does not use any complex data structures, it process the transactions directly. It proceeds by recursive elimination. Firstly, it selects all transactions that have the least frequent item (among those items that are frequent). Then, it removes that item from them and recursively process the reduced database remembering the items found during the recursion. When the recursion ends, all frequent item-sets for the removed item have been computed.

Table 2.1: Taxonomy of the related work

Related Work	Hierarchical Items	Real-Time	Dimensionality
FIM	√	-	High
HHH	√	√	Low
AutoFocus	√	-	Low

Table 2.2: Itemset reported after a DDoS attack from multiple source IPs towards IP A and port B

SrcIP	dstIP	srcPort	dstPort	proto
*	A	*	B	*

Afterwards, the algorithm repeats the process with the second least frequent item and without the already processed item in the database.

SaM [45], purely based on horizontal representation, is a simplified version of RElim. It performs in two steps: split and merge. In the split step, all arrays starting with the leading item of the first transaction are copied into new arrays and that leading item is removed. This process is repeated recursively to find all frequent item-sets for the leading item. Then, in order to obtain the conditional pattern base not containing the leading item, a merge step with the rest of the database not containing that item is needed. Optimized versions of both RElim and SaM have been recently proposed by the authors [47].

In Table 2.1 we summarise how the main previous works differ in the dimensionality of the input records, the type of items (flat or hierarchical), and the (near) real-time or offline processing of the input records.

2.4.1 FIM for anomaly detection

The main idea behind using frequent item-set mining for anomaly detection is the fact that a security anomaly such as a scan, a network scan or a DDoS will imply that there will be a significant amount of flows sharing a certain subset of the flow features. For instance, if there is a DDoS towards IP A and port B, FIM would find an item-set as the one we can see in Table 2.2. As we can observe, while the dstIP and dstPort columns have specific values, the remaining all contain wild cards, which stand for non specific values. The item-set showed in the table summarises in a compact way all flows with destination IP A and destination port B.

In frequent item-set mining for anomaly detection, the minimum support is specified in terms of flows (each transaction is a flow). It will indicate the amount of flows that

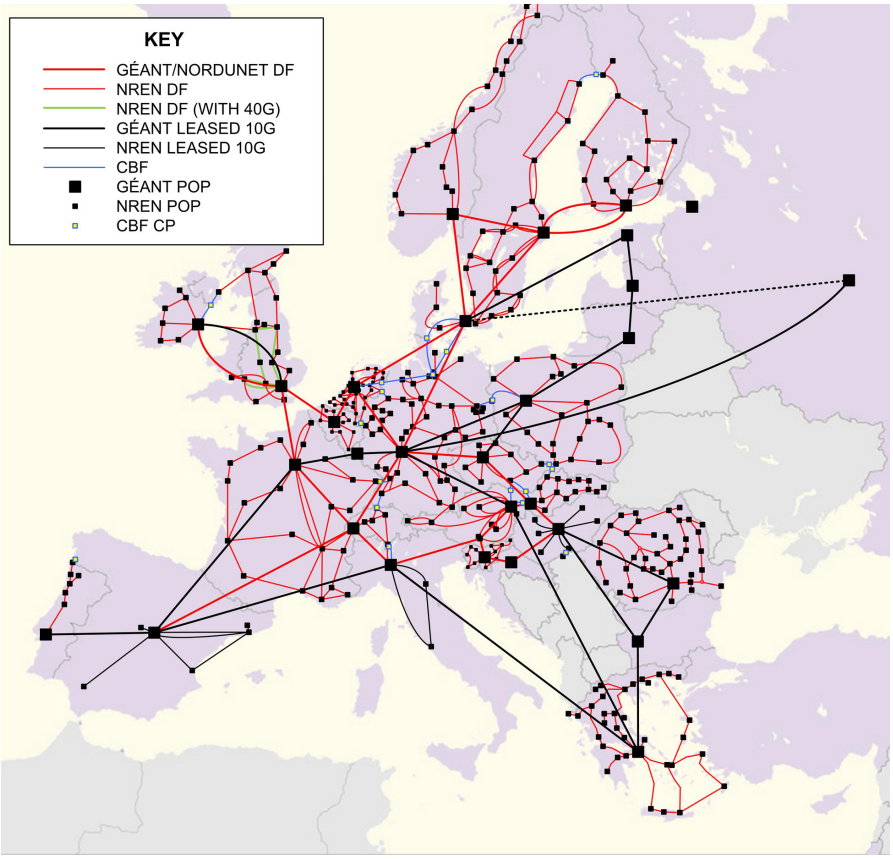


Fig. 2.1: GÉANT network.

an attack must have in order to be reported as a frequent item-set by the miner. The fine tuning of this parameter is difficult and depends on several factors. Namely what attack wants to be detected, the characteristics of the network where it is being tested or the presence of sampling. While a low s will lead to many false positives (frequent item-sets corresponding to legitimate traffic), a high value might reduce significantly the detection rate by significantly increasing the false negatives (anomalies going under the radar).

2.5 Scenarios

In this section, we present the three different network scenarios that have been used to perform the experiments of this thesis.

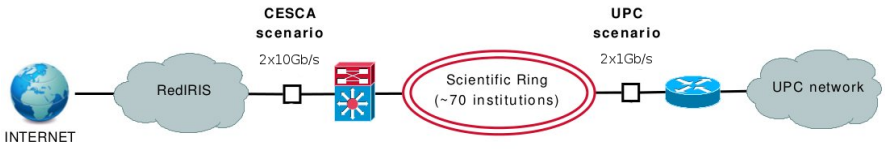


Fig. 2.2: CESCA and UPC scenarios.

2.5.1 GÉANT scenario

DANTE [48] is a non-profit organization that plans, builds and operates the GÉANT [49] backbone network. GÉANT is a /19 transit network connecting 34 European NRENs with 18 points-of- presence (PoPs) spread over Europe (with 10Gb/s links almost everywhere), a dozen of non-european NRENs, and two commercial providers (Telia an Global Crossing) (see Figure 2.1). It is the main interconnection point for inter-NREN traffic. For a certain subset of NRENs, GÉANT is also the primary gateway to the commercial Internet (other NRENs have their own connection to the non -research world). Although it is a R&E network, more than half of the traffic is towards commercial providers. The overall handled traffic is more than 50Gb/s.

DANTE collects Sampled NetFlow [26] from the GÉANT core routers. NetFlow is collected from every router interface with an external peering network. As GÉANT is a purely transit network, this setup is sufficient to account for all the traffic.

2.5.2 CESCA scenario

This scenario consists of two 10 Gigabit Ethernet links that connect the Catalan Research and Education Network (also known as Scientific Ring [50]) to the global Internet via its Spanish counterpart (RedIRIS [51]). The Scientific Ring is managed by the Supercomputing Center of Catalonia (CESCA [52]) and connects more than seventy Catalan universities and research centers using many different technologies that range from ADSL to Gigabit Ethernet [50] (see Figure 2.2).

2.5.3 UPC scenario

This scenario is composed by two Gigabit Ethernet links that connect our university, Universitat Politècnica de Catalunya BarcelonaTech (UPC), with the Internet through the Scientific Ring [50]. UPC connects around 10 campuses, 25 faculties and 40 departments to the Internet through the Scientific Ring (see Figure 2.2).

Chapter 3

Anomaly Analysis

3.1 Introduction

The amount of information on current anomalies happening in backbone networks as well as their characteristics and behaviour is scarce in the literature [2, 3, 4, 5, 6]. For this reason, this chapter addresses this issue by analysing the anomalies found for several months in the European backbone network of GÉANT.

During fall 2008, DANTE analysed three commercial tools for anomaly detection. One year after (fall 2009), one of those tools was permanently deployed in the GÉANT network. This chapter reports on the benchmarking of the tools and the obtained results obtained using the finally deployed tool for half a year. First, it describes the limitations of current commercial tools and discusses some aspects that still need further research from the perspective of a network operator. Second, it provides a long-term study of the anomalies occurring in a continental backbone network.

After manually analysing more than 1000 attacks, we found that, surprisingly, the overlap among the anomalies detected by different tools is extremely low. This is a clear indicator that false negatives are still significant even when comparing commercial tools that are supposed to detect the same sort of anomalies. In addition, our study reveals that *Network Scan* attacks are the most persistent and shows that there are certain geographical regions that are predominant when looking at the top attackers or targets respectively.

The remainder of this chapter is organized as follows. Section 3.2 describes the followed methodology as well as the requirements used by DANTE to build a short-list of suitable anomaly detection tools. Afterwards, Section 3.3 reports on the differences found among those tools during the evaluation phase in terms of usability, true and false

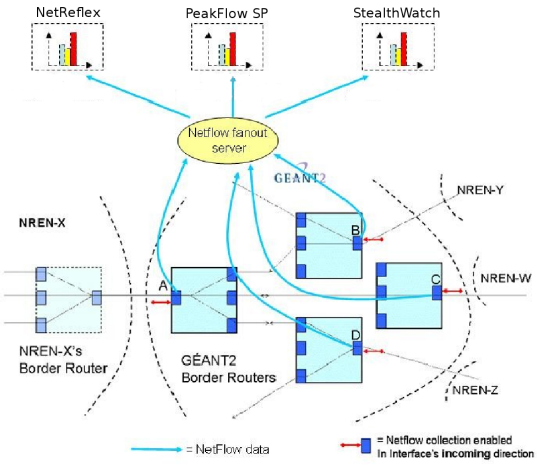


Fig. 3.1: NetFlow collection scenario in the GÉANT network

positives, false negatives and also regarding the different types of anomalies detected. Section A.3 presents a study of the network anomalies found in GÉANT along with their properties after using the selected tool for approximately six months. Finally, Section A.4 summarises and concludes this chapter.

3.2 Context

During fall 2008, DANTE started looking for a solution to enhance the security of its network, and of its customer networks, by analysing three different anomaly detection commercial products. After evaluating the performance of each tool with the same input data for several months, one of them was permanently deployed in the GÉANT backbone network (mid November 2009).

At the beginning of this study, the sampling rate in Sampled NetFlow was set to 1/1000. Later on, the routers were replaced, which allowed to migrate to 1/100 sampling. Therefore, we must take into account that the analysis of the tools presented in Section 3.3 (fall 2008) and the study presented in Section A.3 (2009-2010) were performed under different sampling rates.

NetFlow v5 [7] was used since anomaly detection tools require visibility on very granular flows (the ones defined by the 5-tuple src/dst IP, src/dst port and protocol), which is the default (and only one) provided by NetFlow v5. The NetFlow traffic

is exported to a single fanout box duplicating it towards multiple destinations (see Figure 3.1). This setup allowed us to evaluate all the anomaly detection tools using exactly the same input data.

3.2.1 Tool Requirements

In order to start the process to select one tool for anomaly detection, three candidate tools were short-listed on the basis of a set of requirements. We think that those requirements are representative enough to be useful for any other network operator or company willing to deploy a similar solution. In addition, we think that they impose a serious set of limitations that should be taken into account by researchers working on building anomaly detection algorithms meant to work in real-world networks. Next, we provide the list of requirements along with a brief explanation for each one.

1. Sampled NetFlow support. Given the large scale and traffic volume in backbone networks, one of the main requirements of network operators is the ability of the tools to work with sampled flow-level data (e.g., Sampled NetFlow [26]). Several state-of-the-art tools require access to packet payloads, which renders these solutions impractical for this environment. Recent studies [53, 13, 14] have shown that the accuracy of certain anomaly detection techniques is dramatically affected under sampling.

2. Non intrusive collection of data. Tools require often other data, beyond NetFlow. Specifically, sometimes configuration information about the routers needs to be collected to build a tool representation of the network topology and/or to correlate it with information contained in the NetFlow records (e.g., the interface id). Some tools require collection of BGP data, some also IS-IS[1]. The collection of other data should not impose the deployment of additional hardware or difficult configuration changes in the routers. Since most of GÉANT's customers and peering connectivity points are on 10 Gb/s lines, for cost and deployment complexity, approaches requiring the installation of dedicated probes are not appropriate.

3. Accurate detection and classification. For an operator, it is essential to be able to differentiate the anomaly type (correct anomaly classification), to report the end hosts involved and to detect the anomaly duration. It is also very important to detect both the start and the end time of the anomaly (anomaly window) with a precision in the order of several minutes. Based on DANTE's NOC (Network Operations Center)

[1]BGP (Border Gateway Protocol) is the most commonly used protocol to exchange routing information between Autonomous Systems while IS-IS (Intermediate System To Intermediate System) is limited to an administrative domain or network.

engineers experience, the delay between the true event and its detection should not exceed 20-30 min. and false positives should be low enough in order to be treatable by an operator (e.g., no more than 10-15 anomalies per day).

4. Collection of evidence related to anomalies. Collecting information about the anomalies in a structured way is important to investigate and possibly mitigate the anomalies in collaboration with other CERTs[2]. Relevant information includes: IP addresses and ports, time of the incident and entry/exit network points (both routers and peers).

5. Scalability. The scale of the GÉANT network and the type of traffic posed a significant requirement. At the time of this study, GÉANT had around 10 million unique speaking hosts per day on network with global connectivity, mainly composed of 10 Gb/s links carrying a mixture of research (e.g., grid traffic) and more "ordinary Internet traffic. Thus, the problem was to detect anomalies with a huge number of IPs and large volumes of composite traffic.

Another important requirement taken into account in DANTE's case was to have tool support. For this reason, only commercial solutions were considered.

3.2.2 Analysed Tools

As a result of the above requirements, three commercial tools were short-listed: *NetReflex* [54], *PeakFlow SP* [55] and *StealthWatch* [56]. These three tools represent a good cross-section of current best practice techniques in anomaly detection. Moreover, they are all based on different approaches and, therefore, it will be possible to catch a potentially broader range of anomalies with the same input.

Concerning their working scheme, although we cannot go deep into the internal details about what exact algorithms they use due to the fact that they are proprietary, we give below an overview of their main functionalities and architectures. For further details about each particular anomaly detection method these tools are based on, refer to Chapter 6.5.

NetReflex (NR)

Functionality Overview

[2]A CERT (Computer Emergency Response Team) is a group of experts that takes care of any security-related event threatening a NREN.

NetReflex is a non-intrusive system providing real-time and network-wide visibility. By collecting and processing traffic and routing information, *NR* is capable of performing the following three tasks: topology analysis, traffic analysis and anomaly analysis. The first task focuses mainly on auto-discovering the topology of the network. The traffic analysis task performs real-time inspection of the traffic and, finally, the last task focuses on detecting and classifying anomalies.

Architecture

NR consists of a single physical appliance that integrates all functionalities described above. The system is splitted into the following five core parts: topology analysis, traffic analysis, anomaly analysis, search engine and reporting system. In the first component the system provides information on the topology of the network and other information such as the flows entering or exiting a single PoP or the utilization of a particular link. The traffic analysis functionality computes the traffic matrix. With this information an operator can easily spot the PoP pairs exchanging most of the traffic. In the anomaly analysis component, it reports the anomalies detected along with their type (e.g. DDoS) and all the related meta-data (entering and exiting PoP, source or destination IP, destination port, etc.). The system also provides a search engine, that gives access to the raw NetFlow data and allows the user to perform queries based on the IPs, ports, protocol, entering PoP, etc. Finally, the reporting component provides several types of summaries such as anomaly reports or traffic activity at different levels (e.g., PoP-to-PoP or AS-to-AS).

Anomaly Detection Approach

It uses a technique based on a recent research work [57, 19] that employs Principal Component Analysis (PCA). It applies both volume and entropy metrics along with PCA to discriminate what is normal and what is not. It fuses NetFlow, BGP and IS-IS data and creates a PoP to PoP matrix (18x18 in the GÉANT's case). The PoP to PoP traffic is the elementary unit over which the detection algorithm is run, so that every detected anomaly can be attributed to one uni-directional PoP pair. The fusion of different sources of data and algorithms has several advantages. First, the use of routing data can split the traffic into PoP-PoP pairs and enable the anomaly detection on a level of granularity that is useful for taking corrective actions. Second, PCA allows the automatic compensation of the higher variability of the traffic that some PoP-PoP pairs may "naturally have. Finally, entropy-based metrics enable the detection of low volume anomalies that cannot be detected using only metrics based on the variation of volume.

PeakFlow SP (PF)

Functionality Overview

PeakFlow is a network-wide system that correlates flow data, SNMP and routing information to build logical models and learn what network behaviours are normal. The feedback provided by these models is then used by operations staff to detect and mitigate anomalies, improve network performance and make better decisions for traffic management and capacity planning. The main difference between *PF* and the other two products presented in this work is that this tool is the only one providing protection besides detection. It is able to keep crucial services such as the DNS/web servers running after detecting a threat towards them.

Architecture

It consists of five types of appliances: the Collector Platform (CP), the Flow Sensor (FS), the Business Intelligence (BI), the Portal Interface (PI) and the Threat Management System (TMS). The CP is placed in the backbone or in a peering edge and takes care of collecting the NetFlow data. The FS, which is placed in the client edge, extends network security to the customer. The BI appliance analises the network and reports on its performance (e.g., applications being used). The PI component gives access to the service by providing an interface. It can have multiple instances. For example, in the case of GÉANT, each customer (i.e., NREN) could have its own user interface. Finally, the component taking care of security is the TMS. This part of the software is in charge of detecting the anomalies and applying the proper countermeasures to block them while allowing the flow of legitimate traffic.

Anomaly Detection Approach

This software uses statistical-based and signature-based anomaly detection. Regarding the statistical analysis, it detects anomalies on the basis of variation of traffic volumes. It first creates baseline definitions and then compares the real-time traffic against it to look for abnormal deviations. As for the signatures, it tries to match previously stored patterns with the incoming traffic. Although the statistical base of the anomaly detection of this tool is one of the oldest in the market, the tool has the potential benefit of being easily configurable and using a common Knowledge Base leveraging a quite large installation base. Moreover, several customers (around 50 at the time of the test) provide voluntarily their anomaly feeds to the vendor, who has thus the ability to create new signatures triggering anomalies.

StealthWatch (SW)

Functionality Overview

The *StealthWatch* system provides insight about what applications and services are running in the network, how are they performing, and who is using them. Moreover, it uses behavioural-based analysis to detect security anomalies. Taking into account all this information, it allows IT teams to have more detailed insight and make more reliable decisions for crucial tasks such as incident response, troubleshooting or capacity planning.

Architecture

It is divided into five components: Management Console (MC), Flow Collector (FC), Flow Replicator (FR), Flow Sensor (FS) and Identity (ID). The MC is the user interface through which an operator is able to see graphical representations of what is going on in the network (in terms of both security and usage). The FC takes care of collecting the NetFlow data. The FR component is able to aggregate multiple data sources (e.g., NetFlow, SNMP) in a single data stream and forward it to one or more destinations. The FS is in charge of identifying those applications being used across the network. Finally, the ID part maps any unexpected network event with the user or group of users who caused it.

Anomaly Detection Approach

This system employs behavioural-based analysis. Traffic sent or received by hosts is observed for a number of days (learning phase) and then, the host is classified into the best fitting category according to this profiling (e.g., end host or web server). Deviations from what is believed the "normal" behaviour of the host lead to the triggering of anomalies. This tool requires the manual feed of static BGP prefixes to cluster the IPs in groups. Despite the potential scalability weakness of per-host profiling (see Section 3.3.6), it can be very accurate and precise for detecting sudden anomalous behaviour of single hosts (often related to suspicious of malicious activity). The statistical analysis done in the background is complex. However, the user of this tool has the possibility to easily vary the sensitivity of a host or group of hosts to reduce the number of false positives or to whitelist non-interesting anomalies for the specific scenario where the tool is being used.

Table 3.1: Details for the datasets

Label	Duration	Period	#flows/#packets/#bytes	Sampling
dataset-1	13 days	Nov.'08	1.97G/4.12G/3.21T	1/1000
dataset-2	175 days	Nov.'09-May'10	99.38G/699.95G/576.16T	1/100

3.3 Analysis of the Tools

In this section, we analise the three tools presented in Section 3.2. Firstly, the dataset and the methodology followed during the analysis are explained. Afterwards, an analysis of the true and false positives, the false negatives, the type, and distribution of the anomalies is provided for each tool. Finally, we observe how the origins of the anomalies are split in order to see if any of the tools has any bias in the detection.

3.3.1 Dataset and Methodology

The analysis of the tools is based on a 13 days long dataset from GÉANT collected during November 2008 (days 9-12, 16, 18-22, 23-26) with a sampling rate of 1/1000. We refer to this dataset as *dataset-1* (see details in Table 3.1).

Every single anomaly inside *dataset-1* was manually analysed via access to the raw NetFlow records by a DANTE security team with long experience in network security. Some of the anomalies (especially unclear cases) were double-checked with NRENs to obtain an independent validation. An overall of 1006 anomalies were manually analysed.

Each anomaly was classified either as a true positive (TP), a false positive (FP) or as an Unknown (U). TP means that there was enough evidence to confirm that the event was indeed due to a malicious activity, whereas a FP implies that there was a clear indication that the detected anomaly corresponded to legitimate traffic. An anomaly was classified as an unknown when the security team was not able to reach a conclusion and, therefore, they were not able to confirm if it was just normal traffic (FP) or an actual anomaly (TP). A small sample of false negatives (FN) was also analysed as discussed in Section 3.3.4.

3.3.2 Type of Anomalies

We analysed the type and distribution of the anomalies reported by each tool. We classified all the reported alarms into the security categories described in Chapter 2.

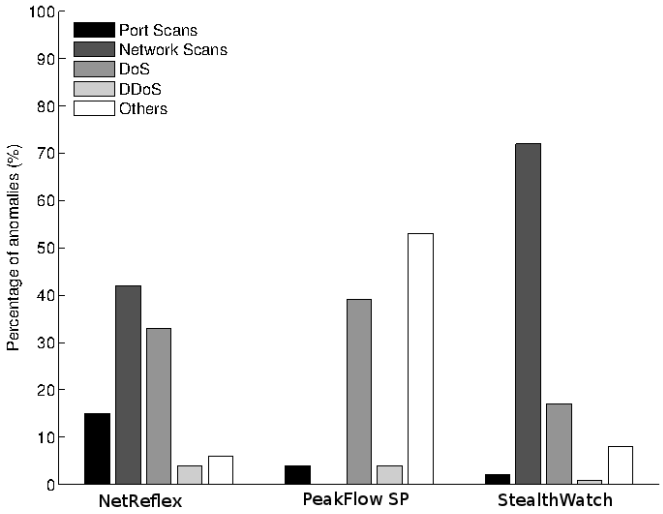


Fig. 3.2: Anomalies reported by each tool

As we can observe in Figure 3.2, NetReflex (NR) and StealthWatch (SW) detected events falling into all categories, while PeakFlow SP (PF) missed completely all the *Network Scans*. This sort of attack is clearly the most frequent one according to NR and SW (42% and 72% respectively). Even though with different percentages, these two tools also coincided classifying the second more frequent attack, the *DoS* (33% and 17%), and the least common, the *DDoS* (4% and $< 1\%$). The most significant discrepancy left was the amount of reported *PS*: while NR detected quite a lot of them (14%), SW only triggered 2%.

PF presented quite different results showing a proportion of 39% of *DoS* attacks and 4% for both *Port Scans* and *DDoS* (respectively). In addition, according to this tool, more than half of the detected anomalies (53%, almost all of them FP, as explained in Section 3.3.4) belonged to the *Others* category while SW and NR showed significantly smaller percentages for that group (8% and 6% respectively). Note that all tools were working under an aggressive sampling rate during the evaluation period (1/1000) and therefore this could have a significant impact on their accuracy.

The results clearly reflect the strong points of the methods the tools are based on. For example, SW, based on per-host behavioural analysis, was the strongest detecting *Network Scans* because when there is scanning activity, the behaviour of a host changes significantly. PF, that uses a baseline to detect abnormal volume variations, was the one detecting more *DoS* because this sort of attack uses large amounts of packets or

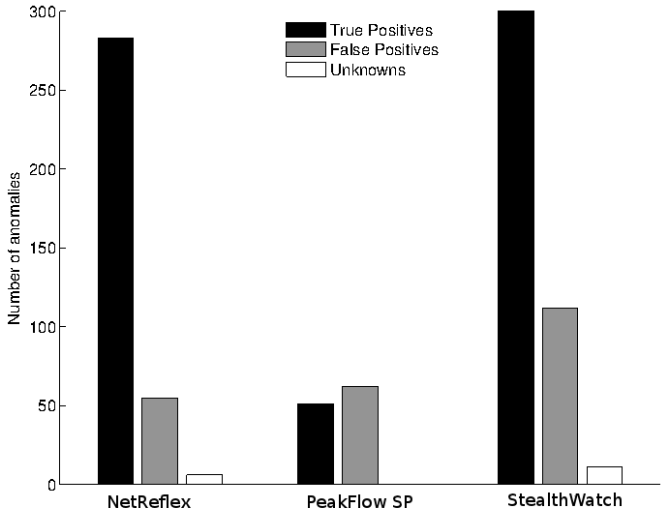


Fig. 3.3: True Positives, False Positives and Unknowns for the evaluated tools

bytes. Finally, NR, based on entropy, showed the best balance among the four types of anomalies.

3.3.3 True and False Positives Analysis

Concerning the figures per tool (see Figure 3.3), SW was the one detecting more anomalies (549) followed by NR (344). PF was the one with the smallest set of reported anomalies: 113. Regarding the True Positives (TP), NR had the best ratio (82.26%) followed closely by SW (77.59%), while PF showed the worst performance with 45.13%. The false positive (FP) ratio for NR was the lowest one (15.98%), while SW had a similar value (20.4%). Half of the anomalies detected by PF (54.86%) were FP. Therefore, NR clearly showed the best ratio TP-FP, although it detected far less anomalies than SW. Regarding the "Unknowns" category, NR and SW had just few cases ($\approx$2% each) and PF did not have any.

Figure 3.4 shows the total number of TP, FP and Unknowns per anomaly type for each tool and also taking into account all tools together. As we can clearly observe in Figure 3.4(a), the amount of overall FP compared to the number of TP was reasonably low for *PS*, *NS* and *DDoS*. On the contrary, the false positive ratio was non negligible in the case of *DoS* (29.2%), while the *Others* category was almost purely composed by FP. Figure 3.4(c) shows that the FP in the former group were basically signaled by PF. The false positives in case of *DoS* were mainly because of NR and SW. For NR

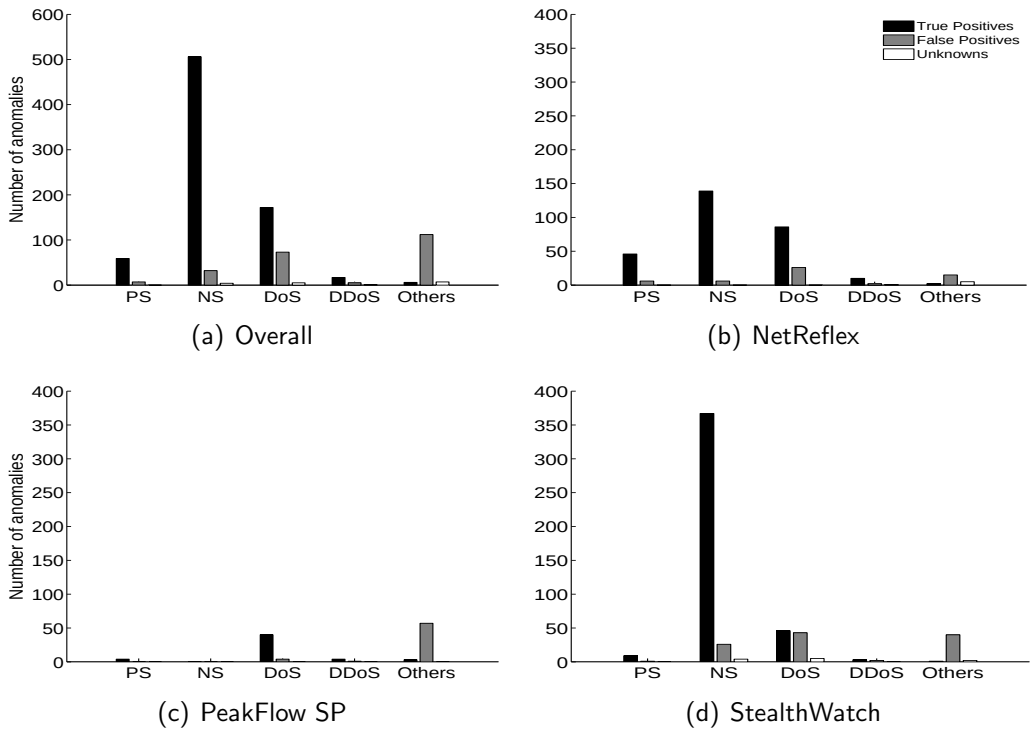


Fig. 3.4: True Positives, False Positives and Unknowns per anomaly type

(Figure 3.4(b)), they were almost one-third of the TP and, for SW (Figure 3.4(d)), the FP were, strangely, as big as the number of true positives.

3.3.4 Anomaly Overlap and False Negatives Analysis

For an anomaly detection system, the tradeoff between false negatives (FN) and FP is very important. Since we knew the set of true anomalies for all tools and also their intersection, we were able to compute a lower bound of the FN for each tool. For a given tool, we know that its lower bound of FN is composed by all TP flagged by the other tools but not detected by the tool itself.

However, analysing the full set of FN was extremely hard in our environment, because it requires manual inspection of every single flow (i.e., 1.97×10^9 flows, not only those belonging to anomalies detected by the tools) to determine whether it is part of an anomaly. We discarded other alternatives, such as performing a penetration

Table 3.2: Lower bound of false negatives per tool and anomaly type for *dataset-1*

	PS	NS	DoS	DDoS	Others	Overall
NetReflex	9	350	67	3	1	430
PeakFlow SP	50	505	122	3	2	682
StealthWatch	40	123	113	12	4	292
Overall	99	978	302	18	7	1404

test, due to legal issues since the analysed sources and destinations were outside the administrative domain of DANTE.

Surprisingly, the intersection among the set of anomalies detected by all tools was limited to a few percent. NR had only 17 anomalies in common with PF and 29 with SW. PF and SW only shared 6 anomalies. This is a strong indication that the particular anomaly detection approach used by each tool clearly influences what kinds of anomalies are reported, even though all tools aim to the detect the same type of events. This highlights the importance of combining different anomaly detection systems to catch a broader range of anomalies.

Concerning the lower bound of false negatives per anomaly type, they all presented huge values as we can observe in Table 3.2 (last row). They were approximately twice as big as the TP for both *NS* and *DoS* (respectively). For *DDoS*, they were almost equal to the TP whereas *PS* showed the lowest ratio (close to two-thirds of the TP). Regarding the overall number of FN for each tool, PF was the one with the highest value (682) followed by NR (430) and SW (292). Note that although all tools detected approximately the same sort of anomalies, the lower bound of false negatives for the tool with the lowest value (SW), already indicates that there were at least 67.91% more anomalies happening in the network besides those being detected by the tool itself.

In order to confirm that the false negatives were so significant, we performed an alternative analysis based on a subset of all the FN. The newly created ground truth of anomalies was manually validated and created independently of the tools. We used *frequent item-set mining* (FIM), which has been recently used in the literature to extract sets of anomalous flows [23, 32, 33, 34, 31, 35] (refer to Chapter 2 for more details). We randomly selected sixteen 30-minute samples of NetFlow within *dataset-1* and run FIM on them. Afterwards, we manually splitted the reported sets of flows into legitimate and anomalous traffic. In case of being anomalous, we also classified them taking into account the types of anomalies described in Section 3.3.2. This final set of anomalies with their corresponding type was our ground truth. Therefore, any anomaly in this ground truth that was not detected by a particular tool was considered a false

negative for that specific tool. We run FIM with the *minimum support (s)* set to 2000 flows, which resulted in a reasonable number of anomalies to be treated manually. Accordingly, this analysis of false negatives was limited to those anomalies reported by NR, PF or SW that had 2000 flows or more. For the analysed intervals of time and having at least 2000 flows, NR, PF and SW had 8, 5 and 4 anomalies respectively.

Although all the anomalies flagged by NR, PF and SW were found using FIM for the analysed periods, we found many more anomalies that had not been detected by any of the three tools. In particular, the manually validated ground truth had 126 anomalies. Therefore, the lowest percentage of FN was for NR (93.65%), closely followed by PF (96.03%) and SW (96.83%). This new evaluation further certified that even current commercial tools are still missing a vast amount of anomalies.

3.3.5 Origin of the Anomalies

As described in Section 3.2.2, SW required the manual introduction of BGP prefixes in order to group hosts and profile their behaviour. To check if that factor had any impact on how SW was performing, we decided to investigate the origin of the anomalies. Figure 3.5 shows that, while the sources of the anomalies detected by PF and NR are spatially split among NRENs and no NRENs, SW shows a bias towards NREN origins, the ones for which DANTE was able to provide the BGP prefixes (NRENs are DANTE's customers). The reason behind this is that DANTE could not create those prefixes for its commercial peers due to their enormous variability and size. Therefore, SW failed to profile those sources and detect anomalies coming from there. The sources labeled as *Mix* stand for those attacks coming from multiple origins that could not be classified either as NRENs or no-NRENs, because they were a mixture of both of them. However, this group was not significant enough to change the bias showed by SW. When we were not able to determine the location of an anomaly in particular (e.g., due to IP spoofing), we classified it as an *Unknown*. This group was not significant either for any of the tools.

3.3.6 Configurability, Scalability and Usability of the Tools

Besides analysing the performance of the tools, network operators are also interested in other important features, such as their configurability, scalability and usability. This section shows a qualitative analysis of such features because we think this can be almost as important as the evaluation itself for operators or any other organization with a large network willing to deploy an anomaly detection system in a similar scenario.

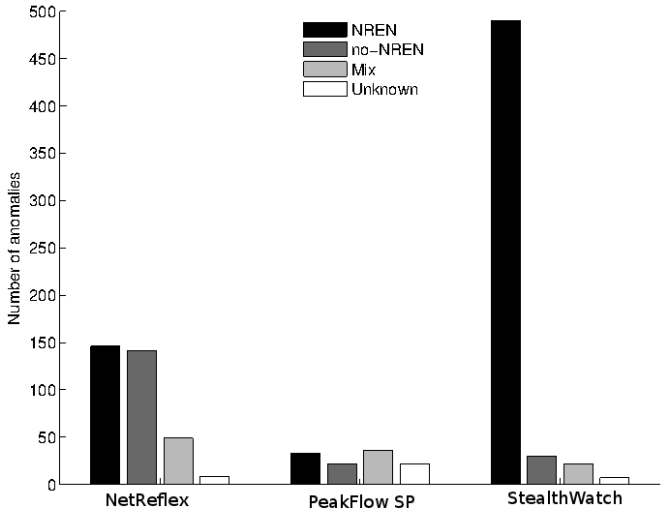


Fig. 3.5: Source of the detected anomalies

We describe below what we learnt during the configuration phase of the tools as well as how usable they were during the evaluation process.

Configurability

NR and PF were running on one server each, whereas SW required a main server and management workstation. All three solutions were deployed in a central location within the GÉANT network (Frankfurt). All tools met requirement 1 (see Section 3.2.1), i.e., they worked with the already existing NetFlow scenario of GÉANT. PF and NR required also SNMP access to the routers for obtaining network configuration information. NR and PF required as well receiving live BGP feeds from our routers. We configured them both to be part of the iBGP (internal BGP) full mesh of our 18 routers. Note that NR uses BGP to build a POP to POP traffic matrix (with NetFlow only it is possible to derive the ingress POP, but not the egress one). PF only used BGP information for doing traffic peering analysis. As explained in Section 3.2.2 and 3.3.5, SW required the manual feed of static BGP prefixes to cluster the IPs of observed traffic in groups. That was easy for DANTE's customers (European NRENs), whose prefixes are fixed or vary with a very low dynamic, but not feasible for the rest due to their large size and higher dynamism. NR also required the collection of IS-IS data, and this was achieved by simply letting the tool server be layer 2 adjacent with one of the routers. This

information was not used for the Anomaly Detection component.

Scalability

All the tools proved to be able to handle and process the NetFlow and other data feeds they needed. In particular, SW proved to be very accurate for the detection of malicious activity originating/targeting only the set of BGP prefixes defined in advance. However, we estimated that extending it to all our peers (including those providing global connectivity - Telia and Global Crossing) would have required a twenty fold increase in the memory requirements of the tool, thus boosting its cost and impacting its performance.

Usability

Both SW and NR provided a compact exporting of the information for the detected anomalies. SW (being host behavioural based) can also show other anomalies associated to one IP that is the source or target of an anomaly. This functionality is not present in NR. However, NR can precisely associate an anomaly to an entry/exit point of the network, and to an entry/exit BGP peer due to its fusion of NetFlow and BGP data. SW is less precise in this respect, especially when at least one of the anomaly entry/exit points does not belong to a European NREN (which is the most common case). Regarding PF, external third-party tools were needed in some cases in order to investigate an event, which made this tool less usable than the others.

Learning Curve

The interface of NR is very intuitive as it clearly defines the different available sections (e.g., anomaly detection and traffic analysis). The system has a very short learning curve as the drill depth is approximately six clicks. The interface of SW was also very straight forward. The user console is very intuitive, providing multiple paths to investigate an event. At the same time, this could be an issue as the route to the solution could be ten to fifteen clicks before reaching the NetFlow level. PF was the least intuitive tool due to the non-practical graph styles and the way to show events back to the operator, which made it harder to analise each anomaly.

3.3.7 Tool Selection

Taking into account the requirements listed in Section 3.2.1 and after evaluating the three tools, NR was finally deployed in the GÉANT backbone network. Next, we

provide a summary of this evaluation and explain the step-by-step reasoning towards our final selection.

Although it is not possible to determine how it affected their performance, all tools were able to work with sampled input (Sampled NetFlow) (requirement 1). Two of them, NR and PF, required no additional hardware other than a server running the software and no complex changes to the routers were necessary (requirement 2). However, SW needed an extra workstation besides the server and also required the manual introduction of BGP prefixes, which for DANTE's case was only possible for the European NRENs, a subset of all their peers. The performance of the tools in terms of both accurate detection and classification (requirement 3) was surprisingly different and a discriminating factor for the selection process. In terms of true and false positives (Section 3.3.3), NR showed the best results with 82.26% of TP and 15.98% of FP, followed closely by SW. PF was the worst in that respect. Regarding the types of anomalies detected (Section 3.3.2), while the PF did not report any *Network Scan*, both SW and NR flagged anomalies of all types. However, when looking at the origin of the attacks (Section 3.3.5), it was clear that SW showed a huge bias towards those anomalies coming from those previously given subset of BGP prefixes (mainly European NRENs), therefore being far less competitive than NR, which provided precise identification of the anomalies irrespective of the peering type. As regards the collection of evidence related to an anomaly (requirement 4), NR was the one providing the highest detail for a reported anomaly, including entry and exit points in GÉANT and related IPs and ports. Finally, regarding the scalability of the tools (requirement 5), SW was the only one that presented issues. In case all BGP prefixes could have been provided, it would have needed an unrealistic amount of memory.

All in all, due to its easy configuration, its best detection and classification, its independence of the origin of the anomaly, its scalability and its higher detail for a reported attack, NR was clearly the best tool given DANTE's requirements, and, therefore, the software finally deployed in GÉANT.

3.3.8 Discussion

It must be noted that the tool finally deployed in GÉANT, NR, is merely anecdotal. The relevant aspect lies on the followed methodology and the performed experiments during the benchmarking rather than on the final decision, which will always depend on the particular network environment and specific needs.

For instance, for SW, it is clear that the impossibility to provide all the BGP prefixes (for both size and scalability issues), significantly reduced the detection capabilities of

the tool. In a different scenario, where the organization willing to deploy it could provide this information, it could be perfectly possible that SW outperformed NR.

Regarding PF, although it showed the worst overall performance, it was the best tool detecting *Denial-of-Service* attacks, with almost no false positives (both SW and NR reported significant proportions of FP for this particular anomaly). Moreover, PF is the only solution providing mitigation of an attack after detection. To give an example, for an ASP (application service provider), which is interested in assuring high quality and availability of its services, this solution would fit better than the others because it flags *DoS* with high accuracy and, additionally, is capable of blocking the malicious traffic while allowing legitimate users to continue using the service.

Please note that, even though PF performed poorly in GÉANT, according to its manufacturer, Arbor Networks, PF is one of the most widely deployed commercial solutions for anomaly detection. Therefore, this confirms the fact that a particular software is neither good nor bad by itself, but depends on its adequacy to the network environment and the singular requirements of the operator.

3.4 Analysis of the Anomalies

After selecting the set of tools to analise (Section 3.2) and evaluating them (Section 3.3), NetReflex (NR) was deployed in the GÉANT backbone network. In this section, we present a study about the anomalies we have found after operationally using this tool for half a year. We show their their types, properties, magnitudes, origins and destinations.

3.4.1 Validation of the Deployed Tool

This study is based on another dataset labeled as *dataset-2* (see details in Table 3.1). While *dataset-1* was obtained during the evaluation of the three tools, *dataset-2* was collected using the deployed tool (NR). *dataset-2* covers almost a seven months period (10^{th} of November 2009 - 3^{rd} of May 2010). In order to confirm that the tool was performing as expected, since it was not feasible to manually check all the anomalies in *dataset-2* due to its duration, a subset of six days of NetFlow data (10^{th}, 11^{th}, 16^{th}, 17^{th}, 23^{rd} and 26^{th} of November 2009) was collected (see details in Table 3.3). All these anomalies were manually validated following exactly the same methodology explained in Section 3.3.1.

In Table 3.3 we can observe that the TP-FP (88.01%-11.99%) of NR improved with respect to its TP-FP during the tool evaluation period (82.26%-15.98%, Section 3.3).

Table 3.3: Details for the 6 manually analysed days of *dataset-2* (November 2009).

Day	#TP	#FP	#flows	#packets	#bytes
10^{th}	39	9	619.43M	4.09G	2.15T
11^{th}	40	4	613.06M	4.01G	2.10T
16^{th}	44	4	581.34M	3.55G	1.99T
17^{th}	38	2	604.66M	3.73G	2.06T
23^{rd}	45	9	598M	3.45G	2.03T
26^{th}	29	4	560.06M	3.67G	2.01T
Overall	235	32	3.57G	22.5G	12.33T

The main reason behind this difference are two key modifications made to NR after its deployment in GÉANT. In particular, the changes made by the vendor were the following. Firstly, the traffic in case of UDP floods was systematically checked in the reverse direction of the attack (i.e., outgoing from the target). This check was motivated by the fact that, during the evaluation period, it was observed that most of the false positives for *DoS* were actually bandwidth tests, large transfers, high-volume P2P activity and data streaming, all of which require minimal bidirectional interaction between the involved hosts. Consequently, if the reverse portion of the traffic turned out to be more than a given threshold of the incoming, it was assumed that there was a legitimate communication going on and no anomaly was signaled. We selected a threshold of 10%, which resulted in a good tradeoff between TP and FP for *DoS*. This change significantly reduced the overall *FP*. Secondly, the sensitivity of the algorithm towards commonly attacked ports was improved, as it will be explained in Section 3.4.2. Also, recall that *dataset-1* was captured under 1/1000 sampling while *dataset-2* was collected with a sampling rate of 1/100, which might contribute as well to the overall performance improvement of NR.

3.4.2 Anomaly Distribution

As already observed in *dataset-1*, *dataset-2* confirmed that *Network Scans* are clearly the most frequent attack with a percentage of 79%, while the rest of the considered security attacks (*Port Scans*, *DoS* and *DDoS*) represent the remaining 21%. It seems hat *NS* are some sort of background activity that is almost always going on looking for open well-known ports to later on exploit some known vulnerability associated to services running on these ports. *DoS* and *Port Scans* are the next most frequent attacks with a quite similar percentage (11% and 8% respectively). The least common are *DDoS* attacks with only 2% of the reported anomalies.

The reason why the percentage of detected *NS* changed so much between *dataset-*

2 (79%) and *dataset-1* (42%, see Figure 3.2) is, as already mentioned in Section 3.4.1, because NR was tuned after its deployment in GÉANT. The vendor of NR added the capability to select, at configuration time, specific destination ports where to increase the anomaly analysis sensitivity. We used that feature to make it focus on frequently attacked ports that were reported by StealthWatch (SW) but missed by NR during the evaluation. Recall that, according to Figure 3.4, SW detected more than twice as many *NS* as NR (before tuning). In particular, we added ports 22 (SSH), 135 (RPC), 139 (Netbios), 445 (SMB) and 1433 (SQL), which are often misused (e.g., SSH brute force attempts or SQL injections) and thus enhanced the detection of the algorithm.

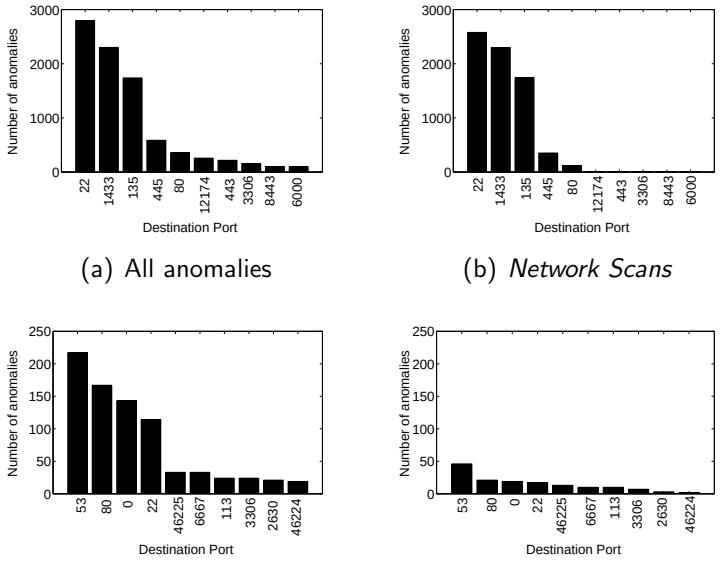
We think that the anomalies detected by NR after the tuning better reflected the reality than before modifying it. The anomaly distribution of NR after tuning (*NS*:79%, *PS*:8%, *DoS*:11%, *DDoS*:2%) is more similar to SW (*NS*:72%, *PS*:2%, *DoS*:17%, *DDoS*:$<$1%). However, note that the distribution of the anomalies found in the GÉANT backbone network could still be biased towards how good or bad is NR in detecting each type of attack.

3.4.3 Top Attacked Ports

Looking at Figure 3.6(a) we can see the top 10 ports attacked taking into account all the anomalies. We observed that seven of them were attacks to well-known ports of widely-known services: port 22 (SSH), port 1433 (Microsoft SQL), ports 135 and 445 (Windows) and port 80 (Web) are in the top 4, while port 443 (HTTPS) and 3306 (MySQL) are in the 7th and 8th position. The other three most attacked ports were 12174, 8443 and 6000. The first port refers to a vulnerability that affects outdated Symantec servers from fall 2009. The second is a popular non-standard alternative for listening to HTTPS connections, and the third is used in X-Window servers. Given that this only shows the top attacked ports regardless of the attack, we then analysed how that distribution looked like for every type of anomaly in order to see if there are particular ports preferred for each kind of attack.

We observed that, as expected, the overall top targeted ports was indeed dominated by the main ports attacked for *Network Scans* (see Figure 3.6(b)), which is expected given that it is, by far, the most frequent attack. The most common subtype is SSH scanning, followed by port 1433 and Windows Netbios ports for networking functions such as file-sharing (ports 135 and 445). Port 80 is the next one, but with lower intensity. Regarding the five ports left (12174, 443, 3306, 8443 and 6000), they all had less than 10 instances during these analysed period.

Regarding *DoS* (Figure 3.6(c)), the most significant attacks happened on ports 53, 80, 0 and 22. Attacks to DNS ports are normally DNS cache poisoning attempts,



(a) All anomalies (b) *Network Scans*

(c) *DoS* (d) *DDoS*

Fig. 3.6: Top attacked ports

which means that a DNS server has received an update from a non-authoritative DNS source and thus its clients are receiving fake data. Regarding attacks to ports 80 and 22, they are quite well-known and intend to saturate either a web or a SSH server to make it unavailable to its clients. We then found port 0, which according to IANA [58] is reserved, but sometimes is used in OS fingerprinting activities. However, we believe that the majority of "port zero" flows are due to packet fragmentation, and to the fact that NetFlow v5 creates a flow with destination port zero for fragmented packets. As showed in Figure 3.6(d), for *DDoS*, port 80 is clearly the most targeted port. The second one is port 6667 (IRC or Internet Relay Chat), which is often used to remotely control hosts previously infected by a Trojan (zombies). These set of hosts are called "botnets" and can be used to launch massive *DDoS* attacks. We finally found the well-known attacks to ports 53 and 22. The set of remaining ports were 25345, 2001 (Trojan), port 0 again and 7000 (Trojan).

Table 3.4: Average number of flows/packets/bytes per anomaly type found in the six manually validated days inside dataset-2

	flows	packets	bytes
Network Scans	1.75K	1.75K	74.8K
Port Scans	153.47K	347.33K	9.64M
DoS	2.33	960.22K	40.10M
DDoS	1.15M	1.15M	46.06M

3.4.4 Magnitude of the Anomalies

In this section, we provide an analysis about the average number of flows, packets and bytes involved in each sort of anomaly. Given that NR does not provide volume information together with the reported anomalies, we used the raw NetFlow data saved for the subset of six days that we manually validated. In order to obtain the correct flows associated to each anomaly, we used the same method described in Section 3.3.4, a recent extension [33] of the Apriori algorithm [23] .

Table 3.4 shows that the volumes associated to each sort of anomaly are significantly different. Concerning the number of flows, it is clear that *DoS* attacks rarely use more than one or two flows, while the other anomalies involve a much higher number. The anomaly type using more flows with a huge difference is the *Distributed DoS*. We then find the scans: in the first place there is the *Port Scan*, with a large number of flows corresponding to the different ports tested and then we clearly observe that the intensity of a *Network Scan* is the lowest (in terms of flows). Regarding the number of packets, the average for *DDoS* and *DoS* is the highest while *NS* are, by far, the attacks using the least. Both *Network Scans* and *DDoS* generate single-packet flows (they show equal amount of packets and flows, respectively). Finally, regarding the number of bytes, *Port Scans* do not seem to be "stealthy activities and have a number of bytes comparable to *DoS* even that being lower. *NS* are the ones using the lowest amount of bytes. Regarding *DoS* and *DDoS* we can see that they look almost exact in terms of packets and bytes but they clearly differ concerning the amount of flows they use for attacking: *DoS* use a few flows while *DDoS* launch attacks coming from a huge amount of sources.

3.4.5 Origin and Destination of the Anomalies

This section presents a study of how the origins and the destinations of the anomalies were distributed over GÉANT. Most of them (56%) were generated from outside GÉANT and 38% of them came from the inside. The remaining 6% came from an un-

known location (see end of Section 3.3.5 for more details). Regarding the destination, a huge amount of the attacks (70%) were directed to the GÉANT network while 24% of the anomalies had outside targets and 6% had unidentified receivers.

Surprisingly, almost half of all the anomalies analysed (45%) came solely from the Asia-Pacific region, specially from China, probably because of the high amount of infected PCs running non-properly patched Windows OS. Regarding the remaining countries, none of them generated (separately) more than 6% of the overall anomalies. Concerning the top targets, there was not a clearly predominant region. Israel (not a EU member but connected to GÉANT) was the most commonly attacked country (8% of the anomalies) along with Greece (7%), North America (6%), Portugal (5%) and Estonia (5%). Israel and Estonia have had and still have some political issues that may explain their presence at the very top even though they are small networks (they receive/send little traffic with respect to other bigger research networks).

When studying the top origin-destination pairs, we found that the most frequent one was from no-NREN to NREN (53.38%). This was due to the fact that the most frequent type of anomaly, as we saw in Section 3.4.2, was the *Network Scan*, which we can consider as some sort of background activity proportional to the number of hosts in a network. Since the no-NRENs represent the "rest of the Internet" (i.e. non-academic networks) it was quite expected this number of anomalies to be numerically dominant. The next pairs were from NREN to no-NREN, with a far lower percentage (21%), and from NREN to NREN (16.78% of the anomalies). The remaining pairs represented less than 9% of the attacks.

3.5 Chapter Summary

In this chapter, we analysed three commercial tools for anomaly detection and provided a study about the type and characteristics of the current security threats happening in a large backbone network. We also reported the strengths and shortcomings found while using these tools, as well as the experience and knowledge we acquired during this long process.

After manually classifying more than 1000 anomalies, we learnt that their distribution, as well as the accuracy of each tool, were significantly different. While the true positives were generally reasonable, the ratio of false positives was quite high for all tools. Surprisingly, we found that the overlap between the anomalies detected by different tools was minimal. This indicates that the number of false negatives is still significant even in commercial tools, and shows the importance of combining different approaches to obtain a stronger anomaly detection system by potentially catching a

broader range of malicious events. As for what tool performed better for each anomaly type, we observed that while StealthWatch (based on host behaviour) was the best with *Network Scans*, PeakFlow SP (based on traffic volumes) was better at discovering *DoS*. NetReflex (based on PCA and entropy) exhibited the best balance regardless of the anomaly type.

In addition, we studied the most common types of anomalies, the top attacked ports, the volumes associated to each anomaly and their sources and destinations after using the deployed tool for approximately six months. Our study revealed the tremendous frequency and persistence of *Network Scan* attacks. We also showed that every type of anomaly had its own preferred destination ports. As expected, the overall top-3 is governed by well-known targets: SSH (22), Microsoft SQL (1433) and a Windows resource-sharing (135). Regarding the magnitude of each sort of anomaly, while *DoS* rarely use more than 1 or 2 flows, *DDoS* attacks generate, by far, the highest amount of flows. *Network Scans* involve the lowest number of packets and bytes and few flows. On the contrary, *Port Scans* use a quite large number of flows, packets and bytes. Moreover, we observed that the Asia-Pacific region turned out to be the region generating most of the attacks while small countries like Israel or Estonia were common targets.

From a practical point of view, we also reported on the acquired experience during this long process. We realized about the complexity and diversity of the traffic observed in a large academic network. For instance, it is quite common to observe traffic that behaves like a *Denial-of-Service* but happens to be some legitimate research experiment. This fact points out a key learning aspect that should be central for any anomaly detection tool. Each network is different and the detection algorithm must be flexible enough to adapt to it. Moreover, the configuration of the tools as well as their long term scalability are important aspects. To give an example, while StealthWatch would have required substantial upkeep in maintaining the prefix lists, NetReflex did not need anything from a manual perspective.

Chapter 4

Impact of Sampling on Anomaly Detection

4.1 Introduction

In this chapter, we first analise the impact of sampling on two scan detection algorithms. Afterwards, based on the obtained results, we propose a sampling technique that works well for scan detection and is able to work on a per-packet basis, thus using less resources than previous proposals and being able to work online.

First, we analise the impact of sampling on TRW [10] and TAPS [11]. In particular, we evaluate them under four sampling techniques: *Packet Sampling*, *Flow Sampling*, *Smart Sampling* [39] and *Selective Sampling* [17]. Details of this algorithms can be found in Chapter 2. Except for *Selective Sampling*, the other sampling techniques were chosen because they are well-known and frequently used in the literature, specially *Packet Sampling* (e.g., Sampled NetFlow [26] is very extended among network operators, and it is based on *Packet Sampling*). We also included *Selective Sampling*, a recent sampling proposal that targets exactly the kind of flows that are typically responsible of the scans (small flows, i.e., with few packets).

Regarding the scan detection algorithms, both TRW and TAPS performance was vastly degraded due to sampling. While TRW turned out to be almost useless under sampling (except for *Selective Sampling*, in which case it exhibited a quite acceptable performance), TAPS showed to be significantly more resilient. Contrarily to the results reported by previous works on the bad performance of *Packet Sampling* for scan detection [14, 13], in our experiments we observed that *Packet Sampling* outperformed *Flow Sampling* in most of the scenarios. Finally, *Selective Sampling* exhibited the best

overall performance among the evaluated sampling techniques.

Second, taking into account the good results reported by *Selective Sampling*, we propose a new sampling technique called *Online Selective Sampling* that also targets small flows. The main problem of *Selective Sampling* is that it must work offline because it first needs to capture all the flows to later apply flow sampling, since it is not possible to know the flow size in advance. Consequently, the main objective of *Online Selective Sampling* is to sample the same flows that *Selective Sampling*, but, instead of collecting all the packets in the first place and then discarding entire flows, *Online Selective Sampling* will sample on a per-packet basis (it is implementable online, before the aggregation of packets into flows occurs). Results show that *Online Selective Sampling* samples the same traffic that *Selective Sampling* but without capturing all the traffic, thus saving time and memory.

In summary, this chapter makes the following contributions:

- We perform the evaluation using the same fraction of flows and also the same fraction of packets and show that *Packet Sampling* can perform better than *Flow Sampling* under fair conditions.

- We present the first analysis of TRW and TAPS under *Selective Sampling* and show that it is better than traditional sampling techniques.

- We propose *Online Selective Sampling*, a sampling technique that targets the same traffic that *Selective Sampling*. However, our proposal is packet-based, i.e., it does not need to aggregate packets into flows to perform the sampling, thus needing less memory and time than *Selective Sampling* and being implementable in NetFlow.

The rest of this chapter is organised as follows. First, Section 4.2 describes the datasets used for the analysis and the followed methodology. Afterwards, Section 4.3 reports the results obtained on the impact of sampling on scan detection. In Section 4.4, our sampling technique proposal, *Online Selective Sampling*, is described and evaluated. Finally, Section 4.5 summarises this chapter.

4.2 Scenario and Methodology

4.2.1 Datasets

For the evaluation, we use four traffic traces from the UPC scenario (see Table 4.1). The datasets are 60-minute traces from 2012 collected at different times of the day

Table 4.1: Detailed information about the traces used.

Trace	Start Time	Duration	Flows	Packets	Bytes
dataset-1	03:00	60 min.	5.5M	59.5M	39.6G
dataset-2	09:00	60 min.	7.9M	128.3M	95.1G
dataset-3	15:00	60 min.	8.6M	136.2M	101.5G
dataset-4	21:00	60 min.	6.5M	94.3M	67.4G

(morning, noon, evening and night).

4.2.2 Ground Truth

In order to analise what is the impact of sampling on TRW and TAPS, we first need to establish a ground truth of true scanners. We followed the same approach proposed in [11, 13, 14], i.e., we created a super set of scanners. In our case, we run TRW, TAPS, Snort and Bro with with loose parameters. Afterwards, we used frequent itemset mining (FIM) and manual inspection as in [32, 33, 8, 34] to double-check that all the scanners in the output were indeed malicious attacks. For more details on FIM, refer to Chapter 2. After this process, we obtained the final list of true scanners (ground truth) against which we will compare the scanners detected after applying sampling. Note that although we can not guarantee that the obtained ground truth is purely composed by true scanners, FIM increases the reliability of the ground truth with respect to previous works [11, 13, 14] because it significantly reduces the need for human intervention, which might be error-prone .

4.2.3 Methodology

We configured TRW and TAPS with a false positive ratio of 0.01, probability of detection to 0.99, probability of having a successful connection being a scanner to 0.2 and to 0.8 for a legitimate host as recommended in [10, 11]. As in [14, 13], the ratio used by TAPS to detect suspicious sources (k) and the time bin to check it (t) were configured differently for each traffic trace and sampling rate in order to obtain the optimal results. Refer to Chapter 2 for details about TRW and TAPS and their configuration parameters.

Even though it is not possible to configure a router to sample a certain percentage of the incoming flows, previous works have only considered a scenario where all sampling methods receive the same fraction of flows to perform the comparison among techniques [13]. However, most routers only support packet-based sampling (e.g.,

Table 4.2: Percentage of sampled flows given a portion of sampled packets (top) and vice versa (bottom) on *dataset-3*.

%pkts	*PS* %flows	*FS* %flows	*SMS* %flows	*SES* %flows
10%	24.21%	11.08%	0.0062%	78.15%

%flows	*PS* %pkts	*FS* %pkts	*SMS* %pkts	*SES* %pkts
10%	3.01%	9.2%	85.35%	0.66%

Sampled NetFlow [26]). That is the reason why it is important to perform the comparison with the same fraction of packets for all sampling techniques. Moreover, note that the sampling rate for packet-based (e.g., *Packet Sampling*) and flow-based (e.g., *Smart Sampling*) sampling techniques has different meanings. While in the first case it refers to the fraction of sampled packets, in the latter case it indicates the portion of sampled flows. This results in a significantly different amount of sampled packets and flows among the different sampling methods (see Table 4.2). For instance, when sampling 10% of the flows from *dataset-3*, *Packet Sampling* receives 3.01% of the packets and *Flow Sampling* gets approximately 9.2%. Therefore, from the point of view of *Packet Sampling*, the comparison is not fair because it is receiving far less packets than *Flow Sampling*. The reason why the same fraction of flows corresponds to such a small percentage of packets for *PS* is that the probability of sampling packets from the same flow is extremely low because most flows are small. Therefore, sampling more than one packet per flow is very unlikely. In order to make all the sampling methods comparable under fair conditions, in this thesis we performed the analysis considering two different scenarios: the same fraction of packets (*sfp*) and the same fraction of flows (*sff*). In *sfp*, all sampling methods take the same percentage of the traffic in terms of packets. On the contrary, in *sff*, the comparison is performed for the same percentage of flows.

We analise the performance of the algorithms using the following metrics previously defined in [11]:

$$\text{Success Ratio: } SR = \frac{\#true\_scanners\_detected}{\#true\_scanners}$$

$$\text{False Positive Ratio: } FPR = \frac{\#false\_scanners\_detected}{\#true\_scanners}$$

The success ratio *SR* indicates how efficient a particular algorithm under sampling is by computing what proportion of the detected scanners matches these scanners in the ground truth (*#true_scanners*). The false positive ratio *FPR* shows how correct is

that algorithm, i.e., it reports the percentage of misclassified scanners (sources wrongly classified as scanners).

4.3 Analysing the Impact of Sampling on Scan Detection

In this section, we study the impact of *Packet Sampling (PS)*, *Flow Sampling (FS)*, *Smart Sampling (SMS)* and *Selective Sampling (SES)* on TRW (Section 4.3.1) and TAPS (Section 4.3.2) scan detection algorithms. For details about each sampling technique and scan detection method, refer to Chapter 2. Afterwards, Section 4.3.3 discusses what portscan detection mechanism and sampling technique are the best options and also compares our results with conclusions reached in previous works.

4.3.1 Impact of Sampling on TRW

Figures 4.1 and 4.2 report on the performance of TRW under sampling for *sfp* and *sff*, respectively. In particular, they show the average and the standard deviation among all traces for the success ratio and the false positive ratio under *sfp* (Section 4.3.1) and *sff* (Section 4.3.1), respectively. Four major conclusions can be extracted from this analysis. First, the performance of TRW is poor even when there is no sampling. Specifically, the amount of reported false negatives is high. Second, the impact of sampling is severe on TRW for all sampling methods except for *SES*. Third, *PS* is better than *FS* for *sfp*. Finally, the *FPR* is small for all the sampling techniques.

Same fraction of packets

Figure 4.1 shows the impact of sampling on TRW for the same fraction of packets (*sfp*). First of all, note that TRW is only able to detect $\approx 20\%$ of the scanners in the ground truth when there is no sampling, which highlights the fact that its low performance is mainly caused by the algorithm itself and not only because of sampling. Moreover, we confirm that sampling further degrades TRW's accuracy as previously reported [14, 13]. Regarding the sampling techniques, we can see that except for *SES*, the impact of sampling on TRW's success ratio (SR) is severe. For instance, under *SMS*, TRW's SR reaches zero with more than 75% of the packets sampled. This is due to the fact that, while TRW tracks single SYN-packet flows to spot scanners, *SMS* samples large flows, i.e., looks down on small flows and, therefore, keeps flows that are useless for TRW. For *PS* and *FS*, the SR degrades linearly for increasing sampling

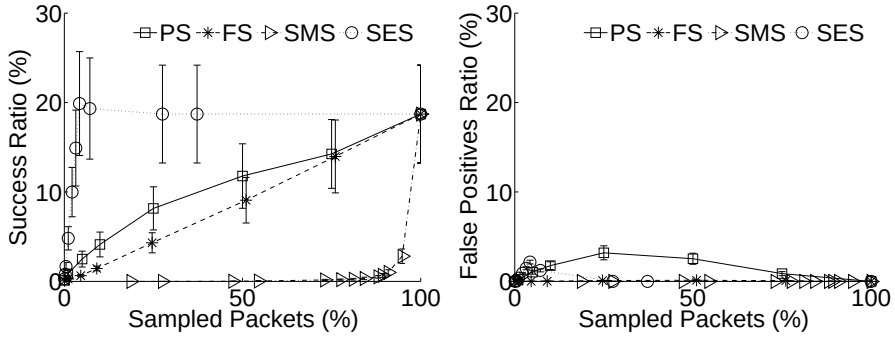


Fig. 4.1: Impact of sampling on TRW (mean $\pm$ stdev). Success Ratio (left) and False Positive Ratio (right) for varying fraction of sampled packets (*sfp*).

rates (s) even though *PS* is slightly better than *FS*. Note that previous works [13] reported that *PS* was worse than *FS*. However, the analysis was performed under *sff*, i.e., under unfair conditions for *PS* (see Section 4.2.3). Therefore, this result shows that it is not true that *FS* is better than *PS* or vice versa. It essentially shows that the final conclusion depends on the metric you use to compare both methods. In contrast to the other sampling techniques, for *SES*, the SR shows to be equal to the unsampled case and gets even higher for s up to $\approx 1\%$. Moreover, for lower sampling rates, *SES* is still capable of detecting some scanners. The reason why TRW performs so good under *SES* is because this sampling method focuses on small flows, which are precisely these flows that TRW looks for to find out scanners. *SES* outperforms the unsampled case because it drops non-small flows, and, therefore, leads TRW to a biased scenario where most of the hosts are only generating single SYN-packet flows. The advantages of opportunistic flow-based sampling techniques with respect to to random methods or even unsampled scenarios have been recently reported in [18].

Regarding the false positive ratio (FPR) under *sfp*, all sampling techniques behave similarly, i.e., they all report low false positives. In particular, *PS* is the sampling technique performing the worst. In particular, the highest peak ($\approx 3.2\%$) happens when sampling 25% of the packets. This is due to the well-known *flow-shortening* effect [14, 13], which transforms multi-packet flows into single packet flows and thus leads to the wrong classification of many hosts. Similarly, *SES* presents an unrealistic scenario where most of the hosts are only generating single SYN-packet flows. However, *SES* manages to keep a lower FPR for all sampling rates (its maximum value is close to 2%). Both *FS* and *SMS* show almost no false positives because the former keeps the flow size distribution and the latter leads mainly to false negatives (low SR), but

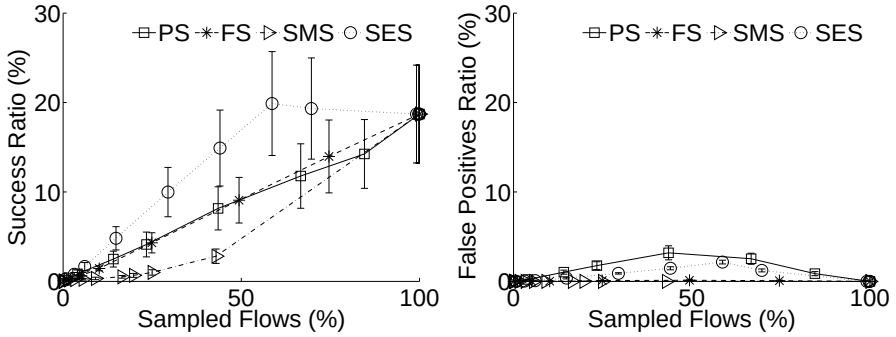


Fig. 4.2: Impact of sampling on TRW (mean $\pm$ stdev). Success Ratio (left) and False Positive Ratio (right) for varying fraction of sampled flows (*sff*).

not to false positives due to the systematic drop of small flows.

Same fraction of flows

Figure 4.2 shows the impact of sampling on TRW under *sff*, i.e., using the same methodology that previous works [13]. Similarly to the *sfp* case, *SES* continues reporting the highest success ratio, *PS* and *FS* show lower and similar performance among each other and, again, *SMS* is clearly the worst sampling method for TRW. Nonetheless, some differences can be spotted with respect to sfp. For instance, the point where *SES*' SR reaches its peak is now at $\approx 30\%$ of sampled flows while in sfp was around 1% of the packets. This 30% of the flows represent only 1% of the packets (they are essentially single packet flows and thus they account for a little amount of the total packets). Also, *SMS* now reaches $SR \approx 0$ around 12% of the flows while for sfp it was for $\approx 90\%$ of the packets. This happens due to the fact that *SMS* retains large flows with higher probability and, therefore, a minor part of all flows ($\approx 12\%$) account for almost all packets ($\approx 90\%$). Moreover, with respect to *sfp*, in this scenario the SR reported by *FS* is either better or equal to *PS*'s due to the fact that under *sff*, *PS* receives less packets.

As regards the FPR for *sff*, similarly to the *sfp* case, the results show very low values. This happens because the percentage of sampled traffic is smaller and, therefore, the chances of detecting either true or false scanners are lower as well.

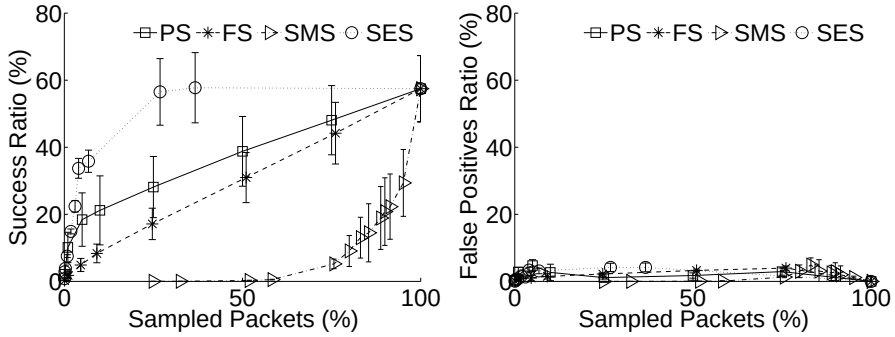


Fig. 4.3: Impact of sampling on TAPS (mean $\pm$ stdev). Success Ratio (left) and False Positive Ratio (right) for varying fraction of sampled packets (*sfp*).

4.3.2 Impact of Sampling on TAPS

Figures 4.3 and 4.4 report on the performance of TAPS under sampling. In particular, they show the average and the standard deviation among all traces for the success ratio and the false positive ratio under *sfp* (Section 4.3.2) and *sff* (Section 4.3.2), respectively. Overall, we can extract the following conclusions. First, the impact of sampling on TAPS is significant but lower than on TRW. Second, similarly to the TRW case, *PS* outperforms *FS* for *sfp*. Third, *SES* performs far better than the other sampling techniques for *sfp*. Finally, the FPR are low but slightly higher than TRW's.

Same fraction of packets

Figure 4.3 shows the impact of sampling on TAPS for *sfp*. First of all, note that, on average, TAPS is only able to detect 57.47% of the scanners in the ground truth when there is no sampling. Although the detection rate is far higher than TRW's, this highlights the importance of combining several mechanisms to detect a broader range of anomalies (e.g., in our case, using Snort, Bro, TRW and TAPS). The performance degradation among sampling methods is similar to TRW's. Specifically, we observe that *SES* is the sampling method offering the best success ratio followed by *PS* and *FS*, and finally, *SMS*. Similarly to the TRW case, the sampling methods focusing on small flows perfom better than these that take large flows with higher probability (see Section 4.3.1). Moreover, we observe that like for TRW under *sfp*, *PS* reports higher SR than *FS*, which is not aligned with previous works [13].

Similarly to what we observed for both *sfp* and *sff* for TRW, TAPS reports low false positives because...

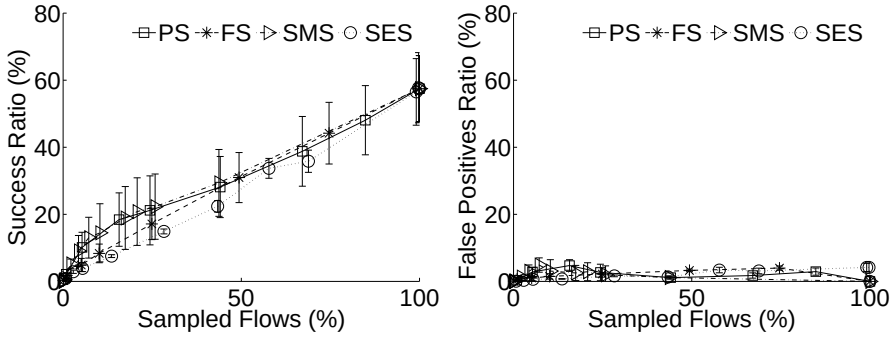


Fig. 4.4: Impact of sampling on TAPS (mean $\pm$ stdev). Success Ratio (left) and False Positive Ratio (right) for varying fraction of sampled flows (*sff*).

Same fraction of flows

Figure 4.4 shows the impact of sampling on TAPS for *sff*. As we can see, the behaviour is quite different with respect to to *sfp*. All sampling methods show a similar degradation in the success ratio for decreasing sampling rate. While *PS* and *FS* already showed such decrease in the SR for *sfp*, *SES* and *SMS* did not. Specifically, this is the first scenario where *SES* does not show any superiority with respect to the other sampling techniques. On the contrary, in this case *SES* turns out to be slightly worse than the other techniques. This is because, for the same percentage of flows, *SES* receives much less packets than the others. Oppositely, for a given fraction of flows, *SMS* takes a very large amount of packets (because it samples big flows). Consequently, the good success ratio showed by *SMS* is due to the fact that it essentially has most of the traffic (e.g., from Table 4.2, only 10% of flows correspond to $\approx 85\%$ of the packets).

Regarding the FPR, the behaviour is similar to *sfp*, i.e., low false positives for all sampling techniques.

4.3.3 Discussion

It is clear that the performance of both TRW and TAPS is severely affected by sampling. However, the TAPS is able to detect more scanners. The main reason behind such behaviour is that TAPS does not depend on any specific packet. While TRW tracks single SYN-packet flows, TAPS does not care about what particular packet of a flow is taken but about the access patterns of each host. This makes TRW more sensitive to the particular packet being discarded. However, although TAPS systematically reports

higher SR, TRW shows slightly better FPR. Overall, as previously noticed [14], TAPS is preferable over TRW in the presence of sampling.

As regards the sampling techniques evaluated, we reach two major conclusions. First, unlike previous works reported [14, 13], *PS* shows better results than *FS* when using the same fraction of packets for both TRW and TAPS (previous studies only used the same fraction of flows to compare the techniques, which was unfair for *PS*). Second, *SES* clearly outperforms the other sampling techniques for all scenarios except for TAPS under *sff*. For this reason, we decided to implement this sampling technique to work on a packet per packet basis (Section 4.4).

4.4 Online Selective Sampling

Many anomalies such as scans use small flows to perform network attacks. Therefore, keeping these flows rather than the large ones facilitates identifying the anomalous traffic out of the whole traffic. There is a recent sampling proposal called *Selective Sampling* [17] (*SES*), whose goal is precisely to take just these small flows (refer to Chapter 2 for details). The problem of *Selective Sampling* is that it first needs to capture all the packets and then samples entire flows, i.e., all incoming flows must be stored in memory until they expire. This working scheme is contradictory with the main goal of traffic sampling, which is precisely to save resources by discarding part of the traffic. Our sampling proposal, *Online Selective Sampling* (*OSES*), targets the same type of traffic (small flows) but instead of capturing full flows, it takes per-packet decisions, thus requiring much less resources and being able to work online.

Recall that *SES* preferentially samples small flows (defined by a threshold in packets) and looks down on large flows. The key idea of our proposal, *OSES*, is to maintain a flow (i.e., to sample its packets) while it is small. If we define a small flow as a x-packet flow ($x \geq 1$), *OSES* will sample all flows having at most x packets with a certain (high) probability and take all the flows having $x + 1$ packets or more with lower probability (the more packets, the higher the discarding probability).

The straightforward solution used in [17] is to first store all flows in a hash table and then, as they expire, remove them from the table with a certain probability (see Section 2.2.5). However, this solution requires a lot of memory and is not fast enough in high-speed links. In particular, the inter-arrival times in links of several Gb/s are in the order of nanoseconds, thus requiring the process time per packet to be incredibly fast [59]. In contrast, we base our solution on bloom filters, which are a feasible option due to their extremely quick look up time and low memory requirements. In our case a flow is kept in the hash table while it is not discarded by *OSES* (instead of waiting

until it finishes or expires as in *SES*).

Next, Section 4.4.1 gives some essential background on bloom filters. Afterwards, Section 4.4.2 explains in detail how *OSES* works and Section 4.4.3 validates its implementation.

4.4.1 Background on Bloom Filters

The idea of a bloom filter [60] is to provide a fast way to decide if an element belongs to a given set. In our case, it is used to control whether a certain flow has been discarded until now. If it it has been dropped before, any incoming packet belonging to it will be directly discarded. Otherwise, the packet will be processed normally. If the new packet forces the flow not to be sampled (e.g., because the flow becomes non small), that flow is immediately set in the bloom filter.

A bloom filter uses a bitmap, which is an array of bits of size m. Initially, all the positions of that bitmap are set to 0. When an element arrives (a new packet in our case) a certain number of hash functions (k) are applied to it. Since we are tracking flows we apply those functions to the corresponding 5-tuple of the packet. Every position referenced by the hash functions is set to 1. According to the expected different elements (flows) to count (n) and the desired false positive probability (p), k and m are set as described in [60]. In order to know if a given element has been already seen, all the positions referenced by its hash functions must have value 1. Otherwise, it means that the element is new. It may be possible (depending on p) to obtain an incorrect answer after looking for an element in the bitmap (false positive). There are no false negatives. As we will see later, a bloom filter needs less memory than a traditional hash table in exchange for a negligible error introduced by the false positives.

4.4.2 Our Proposal: Online Selective Sampling

Online Selective Sampling (OSES) working scheme is described in Algorithm 1. For every incoming packet we first check if its corresponding flow has been previously discarded by looking it up in the bloom filter (line 2). If there is a match, the packet is directly dropped (line 3) because it means that the corresponding flow has been already dropped before. If the flow is not in the bloom filter (line 4), it means that the flow to which that packet belongs to, has not been discarded yet. In this case, the new packet is processed. As it will be explained later, *prob_oses* (line 9) calculates the probability of sampling the current packet. If the returned probability is lower than the random number q (line 10), the packet is dropped (line 13), its flow is deleted from

Algorithm 1: Online Selective Sampling Algorithm

Input: BF: bloom filter;
HT: Hash table of packets;
n: *OSES* parameter;
c: *OSES* parameter;
z: *OSES* parameter;

Output: For each incoming packet, it indicates if it is sampled;

```
1  for every incoming packet pkt do
2      if get(BF, hash_func(pkt)) then
3          discard pkt;
4      else
5          flow = lookup(HT, hash_func(pkt));
6          q = random();
7          x = size_packets(flow);
8          /* probability of sampling the packet */
9          p = prob_oses(x, n, c, z);
10         if p < q then
11             delete(HT, flow);
12             set(BF, hash_func(pkt));
13             discard pkt;
14         else
15             sample pkt;
```

the hash table (line 11) and its 5-tuple is set in the bloom filter (line 12). Otherwise, the packet is sampled.

Since we do not capture the entire flow, we do not know its final size to decide whether it must be sampled or not. We only know its size until the current packet. Since our goal is to sample the same flows that *Selective Sampling* but without capturing all the traffic, we must find a way to make that the probability of taking a flow packet by packet is exactly the same that in the case of sampling the flow directly when all its packets have been captured. If that occurs, both sampling methods would be equivalent. In order to make this happen, the probability for *OSES* must be corrected at each step in such a way that the accumulated probability until the last packet results in exactly the same probability as directly sampling that flow with *SES*.

Consider that p_x^{ses} and p_x^{oses} are the probabilities of sampling a x-packet flow for

SES and *OSES*, respectively. Our goal is:

$$p_x^{oses} = p_x^{ses} \tag{4.1}$$

While for *SES* this probability depends on a single random decision, for *OSES* it is the accumulation of x random and independent decisions, one for each packet of the flow. In particular:

$$p_x^{oses} = \prod_{i=1}^{x} r_i^{oses} \tag{4.2}$$

Consequently, from equations 4.1 and 4.2, the probability of sampling the x-th packet of a flow (r_x^{oses}) can be computed as follows:

$$p_x^{ses} = p_x^{oses} = \prod_{i=1}^{x} r_i^{oses} \rightarrow r_x^{oses} = \frac{p_x^{ses}}{\prod_{i=1}^{x-1} r_i^{oses}} = \frac{p_x^{ses}}{p_{x-1}^{oses}} \tag{4.3}$$

Recall the probability function of *SES* from Chapter 2:

$$p_x^{ses} = \begin{cases} c & x \leq z \\ z/(n \cdot x) & x > z \end{cases} \tag{4.4}$$

In order to accomplish Eq. 4.1, the accumulated probability p_x^{oses} for *OSES* must be equal to c up to z packets and $z/(n \cdot x)$ when the flow becomes bigger. Therefore, for a flow of $x-1$ packets:

$$p_{x-1}^{oses} = \begin{cases} c & x \leq z+1 \\ z/(n \cdot (x-1)) & x \geq z+2 \end{cases} \tag{4.5}$$

Next, we compute the probability of sampling the i-th packet for *OSES* (r_i^{oses}). First, note that for $x = 1$, $p_x^{oses} = r_x^{oses} = c$ to accomplish Eq. 4.1. For $x > 1$, from Eq. 4.3, Eq. 4.4 and Eq. 4.5, we can deduce that the individual probability of sampling the i-th packet of a flow is the following:

$$r_i^{oses} = \begin{cases} c & i = 1 \\ 1 & 1 < i \leq z \\ z/(c \cdot n \cdot i) & i = z+1 \\ (i-1)/i & i > z+1 \end{cases} \tag{4.6}$$

For instance, suppose that we have a flow of size $x = 5$ packets. We configure *SES* with $c = 0.9$, $n = 1$ and $z = 2$. According to *SES*' formula (Eq. 4.4) and these

Table 4.3: Flow size distribution comparison between *SES* and *OSES* on *dataset-1* (c=0.9, z=2, n=1).

Method	1 pkt	2 pkts	3 pkts	4 pkts	5 pkts	6 pkts
SES	73.673%	12.893%	5.319%	3.250%	1.930%	0.99%
OSES	73.672%	12.891%	5.317%	3.241%	1.934%	0.997%

parameters, the probability of sampling that flow is $2/(1 \times 5)$ i.e., 40%. Using the same configuration for *OSES*, the sampling process works as follows.

1. Probability of sampling packet 1:
   - $r_1^{oses} = c = 0.9$
2. Probability of sampling packet 2:
   - $r_2^{oses} = p_2^{ses}/p_1^{oses} = c/c = 1$
3. Probability of sampling packet 3:
   - $r_3^{oses} = p_3^{ses}/p_2^{oses} = z/(n \cdot x \cdot c) = 0.74$
4. Probability of sampling packet 4:
   - $r_4^{oses} = p_4^{ses}/p_3^{oses} = (x-1)/x = 0.75$
5. Probability of sampling packet 5:
   - $r_5^{oses} = p_5^{ses}/p_4^{oses} = (x-1)/x = 0.8$

We can confirm that indeed $p_5^{oses} = p_5^{ses}$. In particular, $p_5^{oses} = \prod_{i=1}^{i=5} r_i^{oses} = 0.9 \cdot 1 \cdot 0.74 \cdot 0.75 \cdot 0.8 = 0.4 = p_5^{ses}$.

4.4.3 Validation

The first goal of this section is to show that the traffic sampled by *OSES* is equivalent to the traffic sampled by *SES*. The bloom filter was configured with $m = 2^{18}$ (the power of 2 is due to implementation reasons). We empirically observed a negligible number of false positives ($<< 1\%$ of the incoming packets). We compare the percentage of sampled flows and packets, the average flow size and the flow size distribution. The average and the standard deviation for several executions on *dataset-1* for $c = 0.9$,

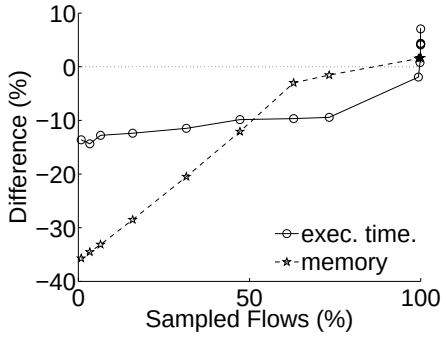


Fig. 4.5: Performance differences between *OSES* and *SES* in terms of both execution time and memory usage on *dataset-1*.

$z = 2$, $n = 1$ resulted in $76.53\% \pm 0.01$ of sampled flows, $11.98\% \pm 0.005$ of sampled packets and an average flow size of $1.68 \pm 8.94 \cdot 10^{-4}$ packets for *OSES*. Similarly, *SES* showed almost identical results: $76.54\% \pm 0.01$ sampled flows, $11.99\% \pm 0.01$ packets and an average of 1.68 ± 0.001 packets per flow. As we can observe, the differences among *SES* and *OSES* are negligible. These minor discrepancies are due to the random decisions. Moreover, Table 4.3 shows that the flow size distribution for both methods is almost identical. Note that for clarity reasons, only the percentage of flows from sizes 1 to 6 packets are reported (the remaining flow sizes account for less than 2% of the total flows all together). Therefore, after this analysis, we confirm that the traffic sampled by our proposal, *OSES*, is indeed equivalent to the traffic sampled by *SES* but without requiring to capture entire flows.

Figure 4.5 shows the resource consumption differences between *OSES* and *SES* on *dataset-1*. In particular, we can observe a percentage that indicates how *OSES* performs with respect to *SES*. For the execution time, the difference is computed as $(t_{OSES} - t_{SES})/t_{SES}$, where t is the execution time. Similarly, for the memory usage, the percentage is calculated as $(m_{OSES} - m_{SES})/m_{SES}$, where m is the maximum memory used. Therefore, while a positive percentage indicates that *OSES* was worse than *SES*, a negative value implies that *OSES* outperformed *SES*. As we can see in the figure, *OSES* is clearly better than *SES* in terms of both execution time and memory usage. For low sampling rates, *OSES* shows its best performance by using almost 40% less memory than *SES* due to the quick drop of a large amount of packets by the bloom filter. Also, *OSES* shows to be approximately 15% faster for the same reason. As the sampling rate gets higher, the differences between both methods are reduced but *OSES* still shows significantly better results. As the percentage of sampled flows

gets closer to 100%, the overhead of the bloom filter in terms of both runtime and memory becomes visible up to the point that *OSES* turns out to be slightly worse (only for more than 99.8% of sampled flows).

The results obtained in this section accomplish our twofold objective, i.e., continue capturing scanners reliably under sampling and using less resources than *SES*. Moreover, while *SES* is not NetFlow-compatible because it is a flow-based sampling mechanism, *OSES* would be easily implementable to work online in most routers because, like NetFlow, it works on a per-packet basis.

Finally, we also evaluated the performance of TRW and TAPS under *OSES*. As expected, similarly to the results reported by *SES* in Section 6.3, *OSES* also showed very good results for scan detection under sampling.

4.5 Chapter Summary

In this chapter we have analysed the impact of sampling on two scan detection algorithms to determine if they are robust enough to continue finding portscans reliably. In contrast to some previous studies, we have been able to see that, under the same fraction of packets, *Packet Sampling* performed better than *Flow Sampling* for scan detection. Furthermore, we proposed a new sampling technique equivalent to *Selective Sampling* that keeps its good results for scan detection but uses less resources and is implementable in the widely deployed NetFlow.

Regarding the analysed scan detection algorithms, we confirmed that both TRW and TAPS were significantly impacted by sampling. However, even though TRW's performance was dramatically affected, TAPS showed higher resilience. Unlike previously reported, we found that *Packet Sampling* outperformed *Flow Sampling* when using the same fraction of packets as the common metric to compare techniques (previous studies had used the same fraction of flows). *Smart Sampling* showed to be rather useless for both TRW and TAPS. In contrast, *Selective Sampling* presented the best behaviour among all the sampling techniques achieving remarkable performance for both TRW and TAPS.

Finally, we proposed *Online Selective Sampling*, a new sampling technique similar to *Selective Sampling* that works on a per-packet basis instead of taking per-flow decisions. Consequently, *Online Selective Sampling* is able to drop flows as soon as they become too large and, therefore, it needs less CPU and memory than *Selective Sampling* while keeping good performance for scan detection.

Chapter 5

Anomaly Detection, Extraction and Classification

5.1 Introduction

In this chapter, we introduce a scheme that automatically detects, extracts and classifies network anomalies. Our scheme combines simple and effective techniques from two worlds: data mining and machine learning.

First, we use frequent item-set mining (FIM) to find a set of *frequent item-sets* (FIs). For more details on FIM, refer to Chapter 2. Second, our scheme builds a decision tree to classify frequent item-sets as anomalous or benign and to determine their specific type in case they are anomalous. A key novelty of our approach is that we classify FIs based on a set of FI features. Intuitively, we decompose observed traffic into distinct groups (FIs) of related traffic flows, which enables us to classify each FI with high accuracy. The decision tree classifies the type of an anomaly along a two-level hierarchy: a main class, e.g., *Port Scans, Network Scans or Denial-of-Service*, and a subclass, e.g., *UDP scan, SYN Scan, ACK Scan*, etc. For detailed information of each anomaly type see Chapter 2. A main advantage of our scheme is that it is conceptually simple and, therefore, easy to understand and configure by a network operator. Our scheme can be used as a stand-alone anomaly detector and classifier or it can be combined with anomaly extraction [23], in which case it only performs anomaly classification.

We have thoroughly evaluated our scheme with real traces and manually verified the detected anomalies. An exhaustive analysis using data from the European-wide backbone network of GÉANT showed that our scheme has very high accuracy (ap-

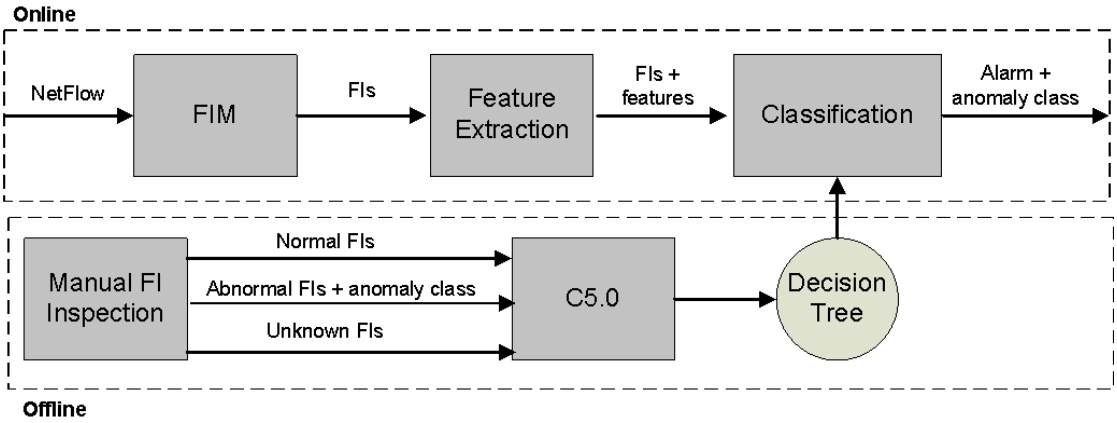


Fig. 5.1: System Overview

proximately 98%). Finally, we have built a corresponding anomaly detection and classification system, have deployed it in a regional academic network and have further confirmed its effectiveness by reporting similar accuracy and low false positive rates (approximately 1%) when monitoring the two 10 Gb/s links that connect the Catalan and the Spanish NREN. The decision trees used by our system were trained with labeled data from the GÉANT network and were found to be accurate when applied to classify anomalies in the regional academic network. This is particularly promising as it shows that it is possible to learn a traffic model in one network and use it effectively in a different network.

The rest of this chapter is organized as follows. Section 5.2 presents our anomaly detection and classification scheme while Section 5.3 describes its evaluation and later deployment. Finally, Section A.4 summarises the obtained results and concludes this chapter.

5.2 System proposal

A fundamental idea of our approach is that the entity that is classified as either normal or anomalous is a frequent item-set (FI). In contrast, previous approaches to network traffic anomaly detection typically classify a time interval based on aggregate metrics derived from the traffic observed during this time interval. We also split time into intervals, e.g., 5-minute intervals, then for each time interval we find the corresponding FIs, and finally we classify FIs. We argue that it is much easier to classify a frequent item-set as normal or anomalous than to classify the entire traffic observed during a

time interval since a FI conveniently groups flows related to anomalies. We describe how we exploit FIM in Section 5.2.1.

Figure 5.1 shows an overview of our anomaly detection and classification scheme. The offline information flow in the bottom of the figure creates a decision tree starting from a semi-manually derived ground truth of normal, anomalous, or unknown frequent item-sets. The ground truth, as we discuss in detail further on, can be derived using the traffic inspection techniques developed in [61] and/or by leveraging information from external anomaly detectors. The normal, anomalous, and unknown frequent item-sets are fed to a learning algorithm, which produces a simple decision tree that is used for detecting anomalies and for classifying the type of a detected anomaly. On top of Figure 5.1, we illustrate the online information flow. Given a set of traffic flows observed, e.g., with NetFlow, during a time interval, we first apply FIM to extract frequent item-sets; then for each frequent item-set we compute a number of traffic features; and finally we use the decision tree to classify frequent item-sets based on their features. We describe our anomaly classification approach based on supervised learning in Section 5.2.2.

5.2.1 Anomaly Detection and Extraction

The extraction of item-sets allows us to represent traffic in a compact form, which more clearly reflects the characteristic behaviour and patterns of network anomalies. The set of items we select to perform the mining on is the well known 5-tuple flow (source/destination IP addresses, source/destination ports and protocol). The selection of these items makes our scheme compatible with NetFlow, which is widely deployed in operational environments.

Table 5.1 shows three example item-sets, one per row. Each of these item-sets summarises a group of flows. Each field can have a specific value or a wild card, which means that there is no restriction for that field. For example, the second item-set summarises all flows coming from one specific IP address that is scanning multiple destination IP addresses on port 80 using different source ports, while the first item-set resembles a distributed attack to a specific destination IP address and port. The learning method we present in Section 5.2.2 automatically identifies the correct anomaly class of an item-set. In our example, it would classify the first item-set as a *DDoS* and the second as a *Network Scan*.

In order to avoid the exponential growth in the FIM utput, we use *maximal item-set mining*. Recall that maximal item-set mining finds only those frequent item-sets that do not have a frequent superset. In the example of Table 5.1, maximal item-set mining would suppress the third item-set. We selected the FPmax* algorithm [62] as a miner,

	Src IP	Dst IP	Src Port	Dst Port	Protocol
Item-set 1	*	1.1.1.1	*	5010	TCP
Item-set 2	2.2.2.2	*	*	80	TCP
Item-set 3	*	*	*	80	TCP

Table 5.1: Example of item-sets

which according to [63] is one of the best performing methods for finding maximal item-sets. Nonetheless, note that our system is not tied to FPmax* and any other frequent item-set miner could be used instead.

Grahne et al. introduced FPmax [64], which used FP-growth to mine only maximal item-sets. Later on, the same author presented FPmax* [62], which, although keeping exactly the same basic algorithm, included some important improvements regarding the quickness of FPmax thanks to the addition of some new data structures.

We specify the *minimum support* in terms of flows, although as we will describe in Section 5.2.2, we also have a separate data structure to deal with attacks that result in a large number of packets, but in a small number of flows, e.g., single *DoS*.

Figure 5.2 shows how the number of frequent item-sets changes with the *minimum support (ms)* parameter. The dashed line shows on the right y-axis the total number of frequent item-sets in the output. The solid line shows on the left y-axis the number of frequent item-sets we have manually identified as anomalous. We observe how these two variables change depending on the *ms*. We observe that although decreasing the *ms* increases the number of anomalous item-sets, this comes at a price: the total amount of found item-sets rises exponentially. Moreover, we observe that for any *ms*, the amount of reported frequent item-sets is always far higher than the number of anomalous item-sets. Our classifier distinguishes between normal and anomalous item-sets. As we describe in Section 5.2.2, we create a ground truth of normal, anomalous, and unknown item-sets and learn their characteristics. This way, our classifier can automatically filter out those item-sets that correspond to normal traffic and reduce the number of item-sets reported.

Compared to the anomaly extraction technique of [23], which applies FIM on a pre-filtered set of flows based on meta-data from an external anomaly detector to extract and summarise anomalous flows, our approach is substantially different. First, we also detect anomalies by classifying FIs using a very simple classifier. Second, we additionally classify detected anomalies. Our scheme can be used as stand-alone anomaly detector and classifier or in combination with the anomaly extraction technique of [23] to classify anomalies. In Section 5.3, we evaluate our system using real traces from two different networks and we show that it obtains high accuracy both with and without an external detector.

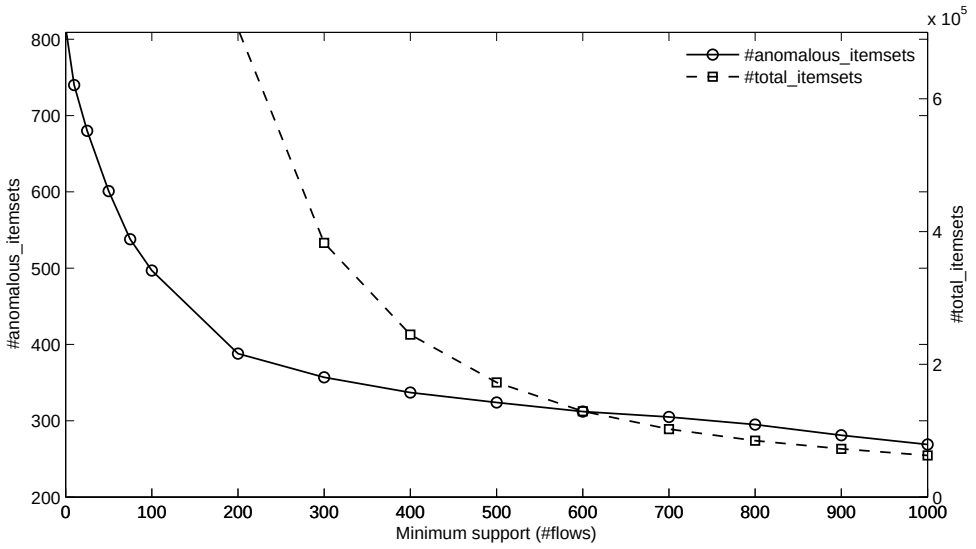


Fig. 5.2: Total number of reported frequent item-sets (right axis) vs. anomalous frequent item-sets (left axis) for varying *minimum support*.

Selecting the correct *ms* parameter requires a small number (typically 2-3) of "trial and error" iterations. As reported in [65], actionable anomaly alarms require on average 60 minutes of investigation, which would correspond to 8 alarms per day assuming a full-time employ for analysing alarms. Therefore, the *ms* parameter should be set high enough so that the output does not overwhelm the administrator of the system with too many alarms. Through extensive experimentation, we have found that a suitable *ms* parameter is typically in the range between 1% and 10% of the number of input flows. In Section 5.3 we use this intuitive rule to guide the selection of the *ms*. For further discussion on the selection of the *ms*, refer to Section 6.3.2.

5.2.2 Anomaly Classification

In order to classify the item-sets extracted by FPmax* we use a machine learning (ML) technique. According to [66], the C4.5 algorithm [67] is one of the methods that offers the best trade-off between execution time and accuracy in traffic classification. We selected its evolution, the C5.0 algorithm.

In order to apply ML for anomaly classification, we first need a ground truth of pre-classified anomalies. Secondly, we have to decide what features will be used to perform the classification. Finally, we need to define a list of anomaly types.

	Value	
	Defined	**Undefined**
Defined Src IP/Dst IP	True	False
Src/Dst Port	Port Number	NaN
Protocol	Protocol Number	NaN
URG/ACK/PSH/RST/SYN/FIN	True	False
Bytes per Packet	$\#Bytes/\#Packets$	
Packets per Flow	$\#Packets/\#Flows$	

Table 5.2: Frequent item-set classification features

The C5.0 algorithm uses a labeled data-set to create a model that predicts the class of an item-set. It selects one feature at each step to split the training dataset accordingly. The importance of a particular feature is given by the *gain ratio (GR)* metric, which is the entropy H for each anomaly class with and without the selected feature. The gain ratio is defined as follows:

$$GR(Class, Feature) = \frac{H(Class) - H(Class|Feature)}{H(Feature)}$$

Establishing a ground truth is one of the most critical phases of any machine learning process, because the entire classification process relies on its accuracy. We have used a set of anomalies detected in the GÉANT backbone network. These anomalies were first detected by three commercial anomaly detectors and then manually validated by the security experts of GÉANT [22]. An important aspect of our classification is that the decision tree needs to distinguish between legitimate and anomalous item-sets. Consequently, we extended the ground truth as described in Section 5.3 with item-sets corresponding to normal traffic.

For each item-set we use the 13 features shown in Table 5.2 that can be trivially derived from NetFlow data. For the source and destination IP addresses, we use a boolean that states if an IP address is part of a frequent item-set. The source and destination port features take values between 1 and 65535. If a port is not defined, a special value is used to indicate that it is not a number (NaN). Similarly, the protocol feature takes a specific protocol value or the NaN value. The *Bytes per Packet (bpp)* and *Packets per Flow (ppf)* features are important to collect information about the bandwidth usage and to capture the used packet sizes, e.g., packets of minimal length are commonly used in attacks such as *Network Scans* or *Port Scans*. Finally, the TCP flags are encoded in six boolean features indicating if a specific flag is set or not.

Our system classifies the anomalies according to the types described in Chapter 2. Note that in the current chapter, the anomaly type *Others* is renamed as *Unknown*. Moreover, there is a new group that includes the legitimate traffic: *Normal*.

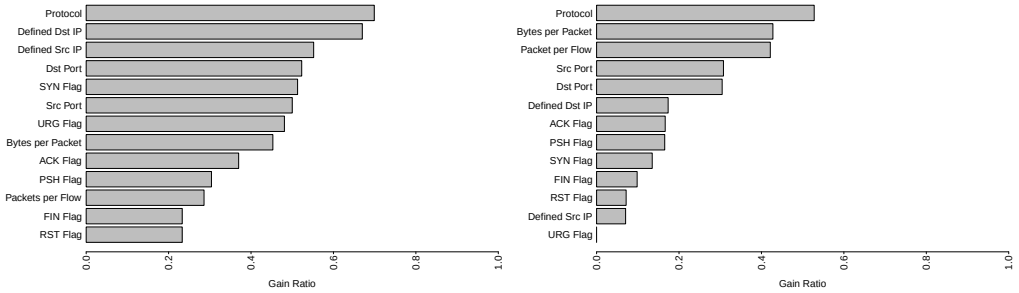


Fig. 5.3: Feature gain ratio for the main (left) and secondary (right) tree.

We created two different decision trees, one for item-sets with a large number of flows (*Main Tree*), and a second for subclasses of *DoS* with a small number of flows (*Secondary Tree*). We split the tree for performance reasons, since the *Secondary Tree* is much more compact than the *Main Tree*.

Finally, we ranked the features with the *gain ratio* metric, which is the discriminator used by C5.0, to understand the importance of each feature and to identify redundant ones. Figure 5.3 shows the that all the features were important except for the URG flag in the *Secondary Tree*.

5.3 Evaluation and Deployment

To build the ground truth for our evaluation, we used a set of 994 anomalies that were first detected by three commercial anomaly detection systems (NetReflex [54], PeakFlow SP [55] and StealthWatch [56]) over a 14-day period and then were manually verified by a team of security experts in GÉANT. For further details on the tools and the analysis refer to Chapter 3 and [22, 8].

We ran FPmax* on raw NetFlow data collected during the reported anomalous time intervals using a starting *minimum support (s)* value of 1000. We manually examined the output to identify item-sets corresponding to the reported anomalies. When we could not find one, we progressively decreased *s* until an anomalous item-set was found. In few cases, which we discarded, the *s* reached almost zero without identifying an anomalous item-set. After this process, we ended up with 760 anomalous item-sets, which we used as ground truth. To classify an item-set as normal or anomalous, we manually examined a sample of the flows of an item-set, e.g., the first 20, using the methodology shown in [33]. We classified an item-set as anomalous, if its flows had suspicious feature values, e.g., destination port 22, and regular patterns pointing to

an automated process, like a bot, that generated them. For example, one such regular pattern is the observation of a very large number of flows with a specific packet size, combination of TCP flags, source addresses, and flow inter-arrival time interval. In order to differentiate between malicious and legitimate traffic, we added normal item-sets to the training set. To find normal item-sets we applied FPmax* on random samples of 30-minute intervals for the same 14-day period during which the anomalies were found. We then manually inspected the flows of approximately 300 new item-sets and classified them as anomalous if they exhibited specific malicious patterns. Finally, since some *DoS* anomalies use a very small number of flows (mostly one or two) but lots of packets or bytes, we also added to the training set item-sets of anomalous and normal traffic with these properties, i.e., a small number of flows and a very large number of packets or bytes. The final ground truth used in the evaluation contains 1249 labeled item-sets.

We use three standard metrics to measure the performance of our classifier. While the first captures its overall accuracy, the second and third metrics evaluate its performance for specific anomaly classes.

$$overall\_accuracy = \frac{TruePositives}{TruePositives+FalsePositives}$$

$$precision(X) = \frac{TruePositives(X)}{TruePositives(X)+FalsePositives(X)}$$

$$recall(X) = \frac{TruePositives(X)}{TruePositives(X)+FalseNegatives(X)}$$

We implemented and evaluated our scheme using data from two different networks: 1) from the European backbone network of GÉANT (Section 5.3.1) and 2) from two 10Gb/s links interconnecting the Catalan with the Spanish NREN (Section 5.3.2).

5.3.1 Evaluation in GÉANT

In these experiments, we used our system solely for traffic classification. To validate the classification results, we used 10-fold cross-validation and report the average figures over 10 rounds. In Figure 5.4 we illustrate the precision and recall per class as well as the overall classification accuracy.

We observe that without balancing, even if the overall accuracy is greater than 95%, our classifier had a moderate performance for certain classes, i.e., note the low precision and recall (two left bars) for *ACK Port Scans* and *ICMP Floods*. Despite the poor performance for these two classes, the overall accuracy with the unbalanced training set was high because these two classes make a small fraction of the overall

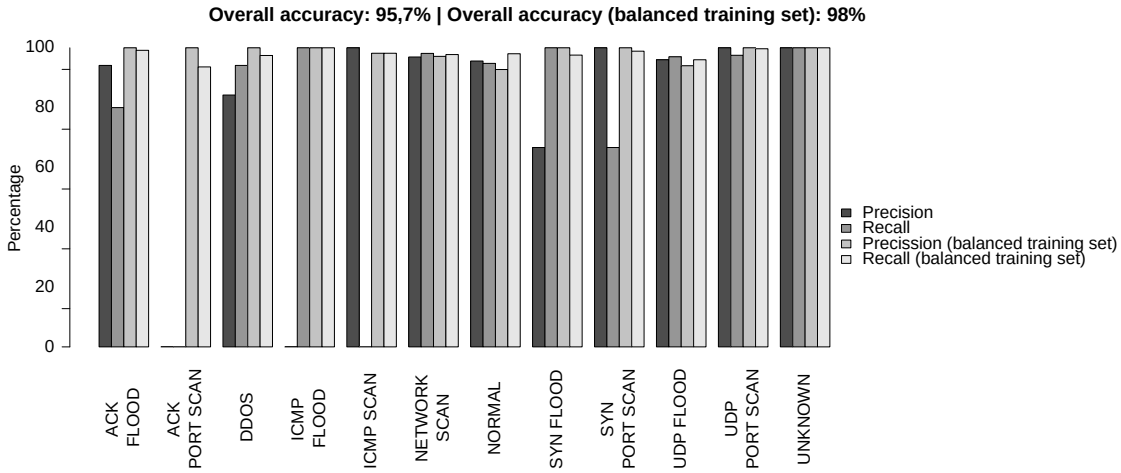


Fig. 5.4: GÉANT classification accuracy

number of anomalies in the training data. To address this problem, we balanced the representativeness of anomaly classes in the training set by increasing the instances of under-represented types. We also balanced the class of normal traffic to increase the frequency of applications with few occurrences (e.g., DNS). With these modifications, our classifier had very good results for the most common anomalies and substantially increased its precision and recall also for the previously under-represented classes. The precision and recall for each anomaly type with the balanced training data is shown by the two right bars in Figure 5.4 and is substantially higher than with the unbalanced training data. The overall accuracy also increased from nearly 96% to approximately 98%.

5.3.2 Deployment in a Production Network Monitoring Platform

In our second scenario, we integrated our system into a production monitoring platform, called SMARTxAC [68], and deployed it in the CESCA network [52]. For details on the network scenario refer to Chapter 2. In the experiments we describe below, we use our scheme for both anomaly detection and classification.

The *minimum support (s)* is set to 3% of the total traffic, which triggers approximately 30 anomalies per day. To classify the item-sets, we used the two decision trees, which were trained using the labeled data derived from GÉANT. To validate the accuracy of our solution, we manually inspected all the detected anomalies during the

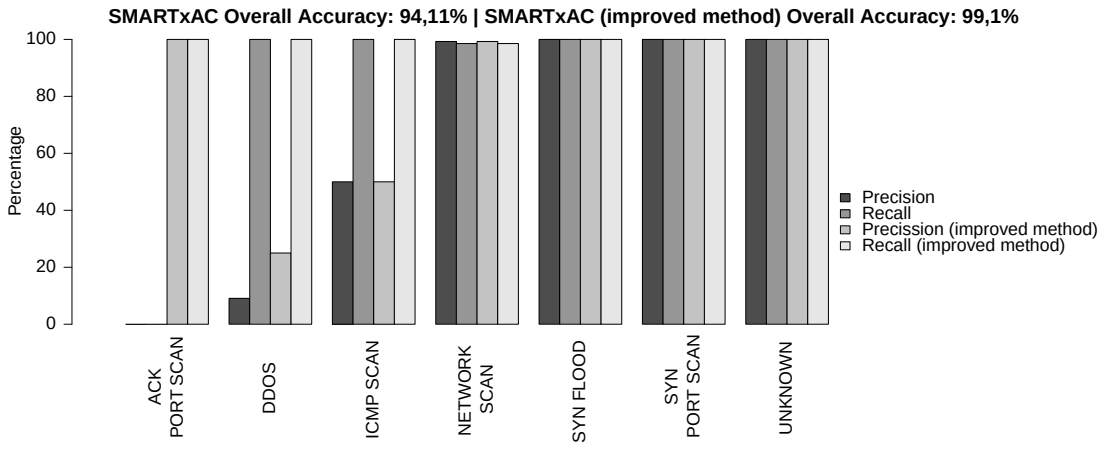


Fig. 5.5: SMARTxAC classification accuracy

first 10 days of August 2011. During this period, the system reported 310 different attacks of 7 different anomaly (sub)classes.

The two leftmost bars of each block in Figure 5.5 show the classification results. The overall accuracy of the system was greater than 94%. For two anomaly classes, i.e., *DDoS* and *ACK Port Scans*, the precision rate was low. These classes made a very small fraction of the total number of anomalies (4,52% and 1,29% respectively). Except for these two cases, the system had a very good accuracy with *only 18 false positives out of 310 anomalies*.

The analysis of the low precision for *DDoS* and *ACK Port Scans* showed that 80% of these false positives were replies from *SYN Floods* and *Network Scans* misclassified as *ACK Port Scans* and *DDoS*, respectively. In order to correct this, we modified the system to look for these specific response patterns. The two rightmost columns of each group in Figure 5.5 show the precision and recall after applying the improvement to the system. As we can observe, the overall accuracy improved to 99,1%. The false positive detection ratio decreased to 1,3% (only 4 erroneously classified anomalies out of 310).

The results in this second scenario show that the decision trees, which were built by learning from the GÉANT network, can be effectively used in a different network. This indicates that the model and the selected features behave well independently of the network characteristics.

5.4 Chapter Summary

In this chapter, we presented a novel scheme to detect and classify network traffic anomalies that combines frequent item-set mining with machine learning. Using real data from two different networks we showed that our solution has a very high classification accuracy, i.e., above 98%, and a low false positive rate, i.e., approximately 1%. In addition, with our proposal it is easy for network operators to understand and reason about detected anomalies. Based on our scheme, we have implemented an anomaly detection and classification system and deployed it in a production network, where it successfully monitors two 10 Gb/s links. Moreover, a particularly promising feature of our classifier is that it has been trained using traffic traces from the European backbone network of GÉANT and has been used successfully to detect and classify anomalies in a substantially different regional network.

Chapter 6

Fast Recognition of High-Dimensional Network Traffic Patterns

6.1 Introduction

In this section, we introduce a fundamentally new approach to extract HHHs based on generalized frequent item-set mining (FIM). Generalized FIM scales much better to higher dimensional data than the well-known AutoFocus [25] and supports attributes of hierarchical nature, like IP addresses and geolocation. We exploit generalized FIM to design and implement a new system, called *FaRNet* (FAst Recognition of high multi-dimensional NETwork traffic patterns), for (near) real-time profiling of network traffic data. Our system is capable of analysing multi-dimensional traffic records with both flat and hierarchical attributes.

We thoroughly evaluate the performance of *FaRNet* using real traffic traces from the European backbone network of GÉANT [49] and show that it scales very well to analysing multi-dimensional data. We find that on commodity hardware that *FaRNet* can process up to 416,000 flows/sec with flat attributes and up to 127,500 flows/sec with hierarchical attributes. We also show that sampling can drastically reduce the memory consumption and CPU overhead, while introducing only a very small error. Compared to AutoFocus, *FaRNet* is much faster, scales better to higher dimensions and produces traffic reports that are more meaningful for a network operator.

A prototype of *FaRNet* has been deployed and used operationally for more than six months in the NOC of GÉANT [33]. We report experiences on how generalized FIM

is useful in practice.

In summary, we make the following contributions:

- We introduce a new approach for discovering hierarchical multi-dimensional heavy hitters based on generalized frequent item-set mining that is much more scalable and flexible than existing schemes.

- We build a new system for network traffic profiling and thoroughly evaluate its performance. We show that *FaRNet* is much faster than AutoFocus. With flat attributes *FaRNet* can process up to 416,000 flows/sec and with hierarchical attributes up to 127,500 flows/sec on commodity hardware. In addition, it scales much better than AutoFocus to higher dimensional data.

- Our system has been deployed and used for more than six months in the NOC of GÉANT, the European backbone network. We describe our experience on how generalized FIM is useful in practice.

The rest of this chapter is organized as follows. In Section 6.2, we describe *FaRNet* and propose modifications to existing FIM algorithms to efficiently deal with flat and hierarchical network traffic data. Section 6.3 shows the results after evaluating *FaRNet* and validating it against the well-known AutoFocus tool. Afterwards, Section 6.4 reports on the deployment of a prototype version of *FaRNet* in a real backbone network. Finally, Section 6.5 concludes this chapter.

6.2 FaRNet: Building an efficient FIM system for network traffic

This section presents our system for FAst Recognition of high multi-dimensional NETwork traffic patterns (*FaRNet*). *FaRNet* efficiently analises network data and extracts useful frequent patterns that summarise the traffic activity of the network. First, Section 6.2.1 presents the architecture of *FaRNet*. Afterwards, we explain how we extend and optimize the FIM algorithms described in Chapter 2 to deal with flat (Section 6.2.2) and hierarchical data (Section 6.2.3). Finally, Section 6.2.4 describes how we use sampling to improve the performance of *FaRNet*.

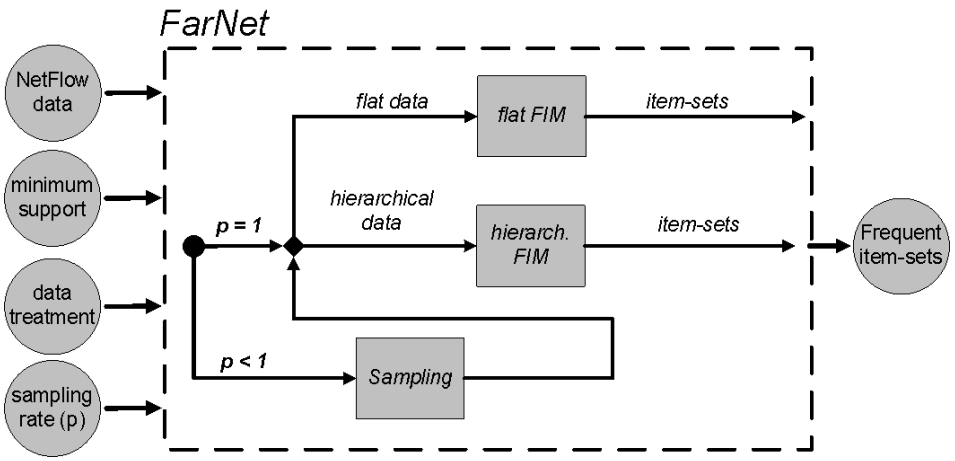


Fig. 6.1: *FaRNet* architecture.

6.2.1 FaRNet Overview and Architecture

Even though previous works (e.g., HH, HHH) are extremely useful to find frequent traffic patterns in network traffic, they are limited to explore a pre-defined set of dimensions (e.g., the well-known 5-tuple in case of AutoFocus). Adding more dimensions results in an explosion in terms of runtime because of the exponential growth of combinations generated. The main objective of *FaRNet* is building a better tool that 1) is capable of dealing with high multi-dimensional data; 2) provides more comprehensive traffic reports; and 3) can perform in a timely fashion.

Figure 6.1 shows the architecture of *FaRNet*. As we can observe, *FaRNet* receives four inputs: NetFlow data, *minimum support (s)*, data treatment and sampling rate (p). *s* is the threshold that determines if the size of a set of flows is big enough to be considered a frequent item-set. The next parameter indicates the type of mining: flat or hierarchical. While for the flat case the input data is considered to be completely plain, in the hierarchical scenario certain dimensions have associated hierarchies. IP addresses consist of prefixes from length 8 to 32 and ports have a two-level hierarchy (specific port and its group, i.e., well-known or not). Similarly, the applications have also two elements, the specific application and its group (e.g., BitTorrent an P2P). Finally, geolocation data has four different elements for each IP (continent, country, region and city). For traffic classification, i.e., to infer the applications, we use machine learning [69]. Note that although our current implementation of *FaRNet* is based on these features, any other hierarchical element could be trivially addded as the system scales well with the number of dimensions. Finally, the sampling rate parameter (p)

Table 6.1: Example of a traditional flow-based report (top three rows) and its equivalent reports for AutoFocus (middle) and *FaRNet* (bottom).

	sIP	dIP	sPort	dPort	proto	app	sAS	dAS	sGeoloc	dGeoloc
Flow-based	1.1.1.1	2.2.2.2	5000	80	6	n/a	n/a	n/a	n/a	n/a
	1.1.1.2	2.3.3.3	6000	80	6	n/a	n/a	n/a	n/a	n/a
	1.1.1.3	2.4.4.4	7000	80	6	n/a	n/a	n/a	n/a	n/a
AutoFocus	1.1.1.0/30	2.0.0.0/8	high-ports	80	6	n/a	n/a	n/a	n/a	n/a
FaRNet	1.1.1.0/30	2.0.0.0/8	high-ports	80	6	HTTP	X	Y	United States	Europe

indicates if sampling will or will not be applied. *FaRNet* has a single output: frequent item-sets.

FaRNet's first step is to sample the input if the user selects a *sampling rate* $p < 1$. Otherwise, it jumps directly to FIM. Depending on the selected type of mining, different paths will be taken. For flat treatment, a FIM algorithm for flat data will be used. For hierarchical treatment, an optimized FIM algorithm extended to deal with hierarchical traffic attributes will be run. This extension, called *Progressive Expansion k-by-k (PEK)*, is explained in Section 6.2.3. The FIM boxes of Figure 6.1 perform maximal item-set mining to suppress redundant information. We adapt, extend and optimize the implementations of different FIM algorithms by [70] to deal with network traffic data.

In Table 6.1, we can observe several examples of network traffic activity reports. The first three lines represent a traditional flow-based report that describes the activity of different hosts accessing distinct servers on port 80. Line 4 shows the output that AutoFocus would return. As we can observe, this report offers more detail by showing the specific prefixes of source and destination IPs as well as the specific range of source ports used. Finally, line 5 shows the output of *FaRNet*. We can see that, additionally to what AutoFocus reported, *FaRNet* is able to find other interesting associations regarding the application used (column 6), the destination AS (column 8) and the geolocation (city, region, country and continent) for both the source and the destination IP addresses (last two columns). From the point of view of a network operator, the usefulness of *FaRNet* with respect to to extensive flow-based reports or the limited case of AutoFocus is clear due to their higher clarity and better summarization of what is truly happening in the network.

6.2.2 FIM with flat attributes

With flat attributes, each flow record corresponds to a transaction with a fixed size of 10 items corresponding to the well-known 5-tuple (source and destination IP addresses,

source and destination ports and protocol), the application, the source and destination AS and the source and destination data for the geolocation. All items are interpreted as plain data. In this case, *FaRNet* uses the FIM algorithms described in Chapter 2 with no further modification. In Section 6.3.2, we compare and evaluate the performance of these FIM algorithms with flat traffic data. Note that the system presented in Chapter 5 for automatic anomaly detection, extraction and classification uses FIM for flat data (transactions have 5 items from the 5-tuple).

6.2.3 FIM with hierarchical attributes

Why is hierarchical mining interesting? For instance, suppose there is a high-volume horizontal scan towards a certain subnet. Although FIM flat would spot the attack, it would only be able to report its source IP and destination port. On the other hand, hierarchical FIM would be able to find the specific subnet under attack and also discover if a certain range of ports had been used (e.g., well-known ports). However, AutoFocus would also find this type of information. Suppose now that there is a large amount of users from different countries in Europe massively accessing websites located in Asia. In this case, neither FIM flat nor AutoFocus would be able to discover such pattern. The former because it does not support hierarchies and the latter because it is limited to the 5-tuple. Nonetheless, FIM hierarchical would find this association in a higher level of the hierarchy, i.e., it would report interaction between Asia and Europe. In hierarchical FIM, IPs, ports, applications and geolocation data are treated as hierarchical elements.

In this thesis, we propose the following three approaches to extend FIM to deal with hierarchical network traffic data: *Full Expansion* (Section 6.2.3), *Progressive Expansion* (Section 6.2.3) and *Progressive Expansion k-by-k* (Section 6.2.3). Next, we present the specific working scheme for each case.

Full Expansion The straightforward solution for allowing FIM to deal with the hierarchical nature of network traffic is expanding each item of the transaction with its corresponding ancestors. For IPs, this means replacing each IP by all its possible prefixes from length 8 to 32, i.e., 25 items (note that prefixes of length shorter than 8 have not been considered since they have never been assigned). Regarding ports, they are translated into two items: its value and its corresponding range. If it is lower than 1024, it belongs to the well-known or low-ports group (0-1023). Otherwise (≥ 1024), it is part of the high-ports (1024-65535). The protocol remains as a plain attribute. Similarly to ports, applications are also translated into two items: the

specific application (e.g., Skype) and the group it belongs to (e.g., VoIP). Concerning the geolocation, for each IP address, we obtain four new elements: the continent, the country, the region and the city. Consequently, in *Full Expansion (FE)* all transactions are extended from 10 items (flat case) to 67 (25 items per IP, two items per port, one for the protocol, two for the application, one for each AS and four for the geolocation of each IP).

Hierarchical FIM will be able to provide greater granularity by automatically finding frequent IP prefixes, interaction between port ranges, associations between continents, countries, regions and cities and the applications generating such traffic. *Full Expansion* results are presented in Section 6.3.2.

Progressive Expansion In *Full Expansion*, all transactions are always extended to 67 items when mining the 10 dimensions. Nonetheless, in most cases, extending all prefixes of an IP is not necessary because a fully defined 32-bit IP is rarely frequent by itself. On the contrary, prefixes of inferior length have higher chances of being above the *minimum support*. For instance, a certain prefix $a.b.0.0/16$ might be frequent even though a more specific subnet (e.g., $a.b.c.0/24$) is not. Similarly, while a city is not often frequent by itself, its corresponding region, country or continent might be.

For simplicity, from here on, all the extensions and optimizations will refer only to IP addresses. However, note that all the proposals made are applicable to the other hierarchical features presented in this thesis and, in general, to any other hierarchical element.

Taking into account that an IP address is rarely frequent by itself, *Progressive Expansion (PE)* will not always generate all its 25 prefixes. An IP prefix of length k will only be explored if its corresponding $k-1$ prefix (parent prefix or ancestor) is frequent. Otherwise, the expansion for that IP will end at level $k-1$. This is because if a certain prefix is not frequent, all prefixes of superior length will not be frequent either (*downward-closure* property, Chapter 2). The frequency of a particular prefix is calculated by progressively counting the frequency of its shorter prefixes (see example in Section 6.2.3). *Progressive Expansion* results are reported in Section 6.3.2.

Progressive Expansion k-by-k The main drawback of *PE* is that it needs to go through all transactions 25 times[1], which is very costly in terms of runtime. Consequently, this section presents *Progressive Expansion k-by-k (PEK)*, which seeks to reduce this part of the process while avoiding the generation of useless prefixes. This

[1]Note that this is because of the depth of the IP address hierarchy and, therefore, it would change depending on the hierarchical element we are dealing with (e.g., 4 for the geolocation).

Algorithm 2: Progressive Expansion k-by-k

```
Input: k: number of bits to expand at each step;
       ms: minimum support;
       Trees: array of trees for /8 prefixes;
1  for l = 8; l ≤ 32; l = l + k do
2      for every transaction T do
3          for every IP address e in T do
4              if l == 8 or parent_is_frequent(e,l - k,ms) then
5                  /*initialize a node if not set*/;
6                  n = get_node_by_prefix(Trees,e,l);
7                  n.count += T.weight;
8                  /*T.weight captures number of bytes, pkts, or flows*/;
9  /*compute unknown frequencies recursively*/;
10 for every tree TR in Trees do
11     if TR.child[left] not NULL then
12         TR.child[left].count = compute_frequencies(TR.child[left]);
13     if TR.child[right] not NULL then
14         TR.child[right].count = compute_frequencies(TR.child[right]);
15 for every transaction T do
16     for every IP address e in T do
17         expand_item(e,Trees,ms);
18         /*walk tree and return frequent ancestors*/
```

is achieved by expanding k bits at each step instead of going one by one (*PE* is a particular case of *PEK* with $k = 1$). When using *PEK*, all transactions will be read $1 + 24/k$ times instead of 25. Note that the only valid values for k are 1, 2, 3, 4, 6, 8, 12 and 24.

Algorithm 2 shows *PEK*'s working scheme. First, all prefixes of length $l = 8$ are generated for all IPs of all transactions and, uniquely for these that are frequent, a binary tree is created (only the root node). Afterwards, for each prefix of length $l + k$ with a frequent ancestor (prefix of length l, tree level $l - 8$), its corresponding tree is expanded up to level $l + k - 8$ (line 6). After going through all possible values of l ($8 \leq l \leq 32$), all frequencies in intermediate nodes (nodes between explored levels, i.e., among $l - 8$ and $l + k - 8$) are recursively computed (lines 12 and 14). Details of the recursivity can be found in Algorithm 3. Finally, transactions are expanded only

Algorithm 3: compute_frequencies

```
Input: n: node of the tree;
1  if n == NULL then
2      return 0;
3  else if n.level == 32 then
4      return n.count;
5  else
6      n.count = compute_frequencies(n.child[left]) +
       compute_frequencies(n.child[right]);
7      return n.count;
```

with those prefixes that are known to be frequent by going though the corresponding tree from the root to the leaves following a depth-first approach (line 17).

The following example illustrates how *PEK* computes the prefixes and frequencies for the IP address $192.168.10.5$ and $k = 2$. The first step consists of generating the binary tree for its prefix of length 8, i.e., $192/8$. Afterwards, if the root node is frequent, prefixes of length 10 are generated (2-bit expansion). Therefore, frequencies for prefixes $192.192/10$, $192.128/10$, $192.64/10$ and $192.0/10$ are calculated. Then, the computation for intermediate nodes (prefixes of length 9) is calculated by moving backwards in the binary tree. In this case, prefixes $192.192/10$ and $192.128/10$ have a common ancestor, i.e., $192.128/9$. Thus, the frequency of the intermediate node $192.128/9$ is the sum of frequencies of its two descendants, $192.192/10$ and $192.128/10$. Likewise, the frequency for $192.0/9$ comes from $192.64/10$ and $192.0/10$.

For $k = 24$, *PEK* would generate all prefixes of length 8 and 32 (same behaviour as *FE* except for the first pruning of infrequent $/8$ prefixes). In this case, *PEK* would be extremely fast (it would go through all transactions only twice). However, it would use a lot of memory (it would directly expand all trees to the maximum of 25 levels without performing any sort of prefix pruning). Therefore, our goal is finding a value of k such that the trade-off between memory usage and runtime is optimal. This trade-off is investigated in Section 6.3.2.

6.2.4 Sampling

In order to reduce the volume of input transactions, we can apply *random sampling*, i.e., we randomly sample each input record with probability p, where $0 < p \leq 1$. Sampling reduces the volume of input data and therefore speeds up the mining process. However,

sampling can have undesired effects. In particular, the frequent item-sets of sampled input may differ from the frequent item-sets of unsampled input. We identify the following four cases:

1. *Identical item-sets*. Both the sampled and the original output yield an identical frequent item-set that has exactly the same items. The flows matching such item-sets are *true positives*.

2. *Lost Item-set*. An item-set that was in the original output does not appear after applying sampling because it is undersampled and its frequency is not above the *minimum support* anymore. The flows belonging to such item-sets are *false negatives*.

3. *New item-set*. An item-set that is not frequent is oversampled and becomes frequent in the sampled transactions. Item-sets with frequency close to the *minimum support* are more likely to transition to frequent. Flows contained in new item-sets are *false positives*.

4. *Transformed Item-set*. This happens when two or more item-sets in the original output are merged into a new item-set in the sampled output. Normally, this new item-set has more defined items than the original because it is precisely a combination of them. In general, all the flows matching this item-set are *true positives* because even though the item-set is not strictly the same, its pattern defines the same set of flows matching the union of original item-sets.

Note that frequent item-sets close to the *minimum support (s)* are more likely to trigger false positives or negatives. The lower *s*, the higher the number of item-sets with frequency close to it. Therefore, the chances of either losing frequent item-sets or creating new ones are higher for smaller *s*.

To estimate the error due to sampling, we model the process with a binomial distribution. In particular, if N is the original size of an item-set, n the size of an item-set after sampling, and p the sampling rate, then the binomial distribution gives the probability of obtaining exactly n successes, i.e., a frequent item-set of size n, out of N independent trials, each of which yields a certain probability of success p. Using the binomial distribution, we can answer the following question. *Given a minimum support s, what reduced s' should be used in order to ensure with high probability that the original frequent item-sets will remain after applying sampling?* Our goal is to find what *minimum support* s' is needed in order to guarantee with a certain probability

Table 6.2: Details of the *dataset*

label	#flows	#packets	#bytes	duration
trace-1	0.51M	5.46M	5.55G	15m.
trace-2	1.96M	25.22M	21.88G	1h.
trace-3	3.87M	47.77M	42.20G	2h.
trace-4	5.77M	72.37M	67.83G	3h.

that an original item-set of N flows will be kept after sampling. Being x the size of that item-set after sampling:

$$P(x > s') = 1 - P(x \leq s') = 1 - CDF(s', N, p) = 1 - \sum_{i=0}^{s'} \binom{N}{i} p^i (1-p)^{N-i}$$

From the formula above, the probability p of an item-set of 1000 flows of reaching $s = 100$ flows after applying a 10% sampling does not even reach 50% ($p \approx 47\%$). However, by using the binomial distribution we can find in advance a desirable setting. In this example, setting $s' = 60$ ensures that the original frequent item-set will not be lost. Nonetheless, there is no need to reduce the *minimum support* so aggressively. Using $s' = 80$ would allow keeping all the original item-sets with high probability ($p \approx 98\%$).

The impact of sampling on the performance and accuracy of *FaRNet* and the trade-off between *true positives* and *false positives* depending on the choice of s' will be discussed in Section 6.3.2.

6.3 Performance Evaluation

In this section, we first describe the scenario and datasets used in the evaluation (Section A.2.1). Afterwards, we report *FaRNet* results with flat and hierarchical data, and also evaluate its performance under sampling (Section 6.3.2). Note that this part of the evaluation focuses on a simplified version of *FaRNet* with a limited number of dimensions (5-tuple) to allow the comparison with AutoFocus [25] (Section 6.3.3). Finally, Section 6.3.4 reports on the results obtained by *FaRNet* when mining the full set of 10 dimensions (5-tuple, application, src/dst AS and src/dst geolocation).

6.3.1 Scenario and Datasets

The experiments presented in this section were performed using four NetFlow traces from 2011 from one of the 18 the points-of-presence of the European backbone network

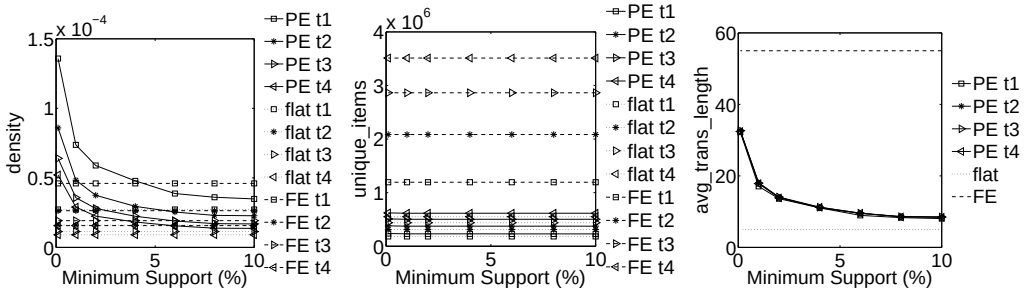


Fig. 6.2: Density (left), number of unique items (middle) and average transaction length (right) for varying *minimum support* on *trace-1 (t1)*, *trace-2 (t2)*, *trace-3 (t3)* and *trace-4 (t4)*.

of GÉANT [49]. For details on the network scenario refer to Chapter 2. The details of the datasets can be found in Table 6.2. Figure 6.2 also plots their average transaction length (l), number of unique items (n) and density ($d = l/n$) for varying *minimum support* (s). As it will be discussed later, these parameters have an important impact on the performance of the mining algorithms, and their values depend not only on the dataset, but also on the particular method applied; i.e., *flat*, *Full Expansion* (FE) or *Progressive Expansion* (PE). For PE, they also depend on the value of s. The figure confirms that traffic data is extremely sparse (only datasets with $d \geq 0.1$ are considered dense in the literature [45]).

6.3.2 FaRNet Performance

In this section, we compare the algorithms presented in Chapter 2 with flat and hierarchical attributes. Afterwards, we select the best performing algorithm among them and show the gain introduced by the optimizations proposed in Section 6.2. Finally, we discuss on the selection of the appropriate value of s. Recall that for all the experiments in this section, only the 5-tuple (i.e., src/dst IP addresses, src/dst ports and protocol) has been used.

We split the four NetFlow traces described in Table 6.2 in time bins of 15 minutes and show the average and the standard deviation in the results. From here on, in this section we will refer to that set of bins as the *dataset*. Note that the longer the time bin considered, the higher the resources needed (i.e., more running time and memory usage). We use an Intel Core2 Quad processor Q9300 and 3GB of main memory running Debian 5.0.5 (32 bits).

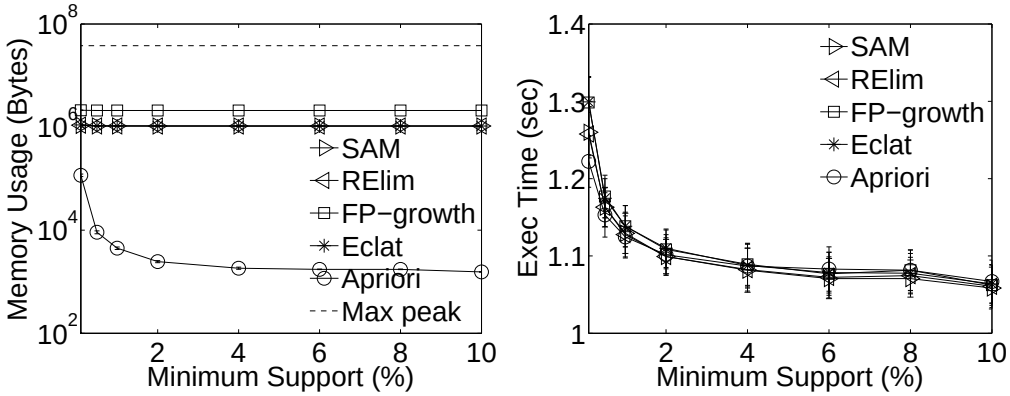


Fig. 6.3: Memory usage (left) and execution time (right) with flat attributes and varying *minimum support* on *dataset*.

Flat treatment

Figure 6.3 shows the performance of all the evaluated FIM algorithms for flat data (i.e., *Apriori*, *Eclat*, *FP-growth*, *RElim* and *SAM*). On the left plot, we observe the memory used for values of s ranging from 0.1% to 10% (we discuss later in Section 6.3.2 on the appropriate selection of s). In order to clearly observe how each algorithm managed memory, the plots show two different values: the maximum total memory used ('Max peak') and the memory strictly reserved during the mining task. The main problem of reporting only the maximum peak is that the specific behaviour of each method is completely hidden below that peak. In particular, the dotted line labeled as 'Max peak' in Figure 6.3 shows the maximum total memory used, which is the same for all the algorithms regardless of s ($\approx 36MB$). This happens for two reasons. First, because the part of the code that takes care of reading the input data and performing a first pass for counting the support of single element item-sets is common to all methods in the code provided in [70]. And second, because this first pass needs far more memory than the rest of the mining process.

Regarding the memory usage of each algorithm, *Apriori* is clearly the one with the lowest consumption. *FP-growth* shows the worst memory usage while *SAM, RElim* and *Eclat* report similar results. The main reason behind such memory differences between *Apriori* and the rest lies on the fact that, except for *Apriori*, all algorithms use complex data structures that book big memory blocks (e.g., ≈1 MB for *SAM*). On the other hand, *Apriori* asks for small pieces of memory every time it needs them.

In contrast, when looking at the execution time of each algorithm (Figure 6.3, right), the differences are hardly noticeable among each other. Overall, after analysing

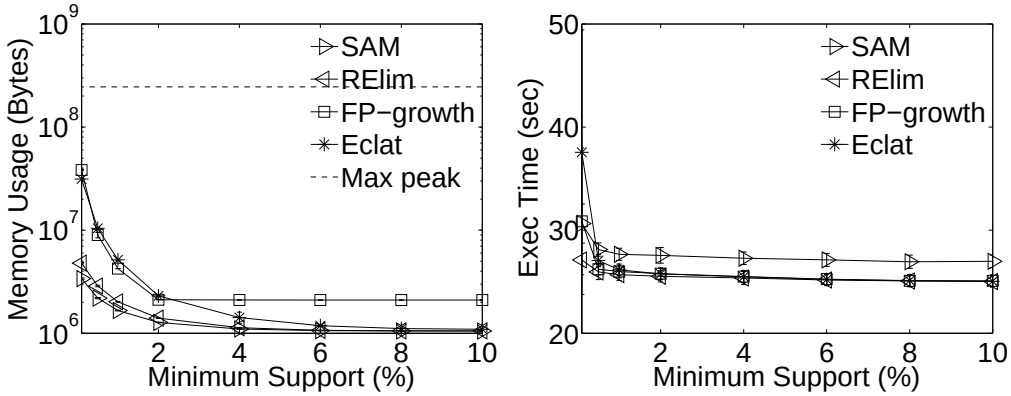


Fig. 6.4: Memory usage (left) and execution time (right) with *Full Expansion* for varying *minimum support* on *dataset*.

both the runtime and the memory used for the mining by each of the algorithms, *Apriori* turns out to be the best algorithm for *FaRNet* for dealing with flat data.

Full Expansion

The objective of this section is to analise how the presented FIM algorithms perform on hierarchical data to, later on, apply the optimizations presented in Section 6.2 to the best performing method. Recall that the goal of mining hierarchical data is to obtain higher granularity on the reported frequent item-sets. For example, if there is a scan, it might be possible to find out the specific subnet that is under attack (with flat treatment this is not possible).

Figure 6.4 shows the performance of *SAM, RElim, FP-growth* and *Eclat* when using *Full Expansion (FE)* on *dataset*. On the left plot, we can observe the memory consumption (in logarithmic scale) for different s values. *FP-growth* and *Eclat* show the highest memory consumption. However, while *FP-growth* is the worst performing algorithm for $s > 2\%$, *Eclat*'s memory consumption becomes close to *SAM*'s and *RElim*'s for $s \geq 4\%$. In contrast, both *SAM* and *RElim* scale smoothly for decreasing values of s (e.g., the memory usage for the mining is approximately one order of magnitude lower than *FP-growth*'s and *Eclat*'s for $s = 0.1\%$).

As regards the runtime, *RElim* and *FP-growth* are the fastest and, for $s < 1\%$, *RElim* is slightly better than *FP-growth*. *FP-growth* is among the quickest algorithms due to its compact FP-tree representation and *RElim* is particularly designed to deal with sparse datasets, which is the case for network traffic data (see Figure 6.2). *Eclat* performs similarly to *RElim* and *FP-growth* but only for $s \geq 1\%$. *SAM* turns out to

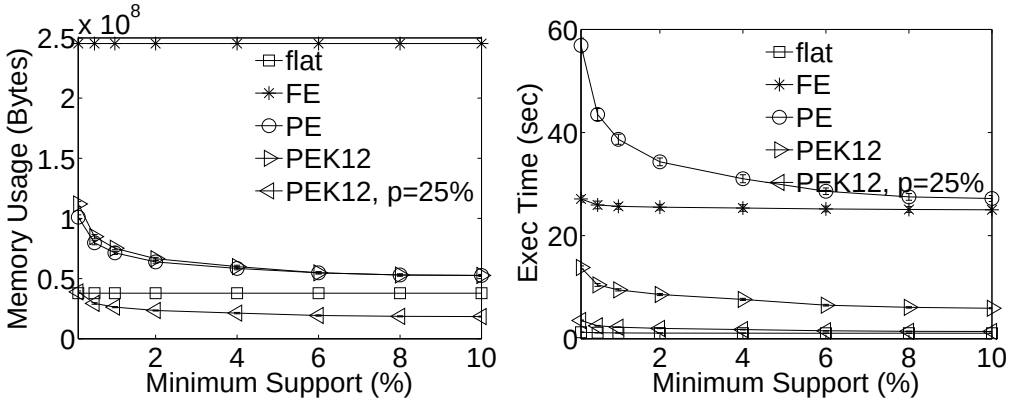


Fig. 6.5: Memory usage (left) and execution time (right) comparison between *flat*, *Full Expansion*, *Progressive Expansion* and *Progressive Expansion k-by-k* for varying *minimum support* on *dataset*.

be the slowest algorithm except for $s = 0.1\%$, in which case *Eclat* shows the worst results.

Note that *Apriori* algorithm does not appear in Figure 6.4 due to scalability issues, which impeded us to run it with hierarchical attributes even for the greatest $s = 10\%$. The main problem with *Apriori* is the candidate generation and testing step, which becomes too slow and needs too much memory with long transactions. Note that in this version of *FaRNet* limited to 5 dimensions (5-tuple), transactions have 55 items in *FE* instead of 5 as in the flat case (25 items for each IP, two per port and one for the protocol).

Based on Figure 6.4, *RElim* shows the best trade-off between execution time and memory consumption for *Full Expansion* in *FaRNet*. Therefore, in the following subsections we evaluate the different optimizations presented in Section 6.2 on top of *RElim*.

Progressive Expansion

In this section, we discuss the advantages of using *Progressive Expansion* over *Full Expansion*. Figure 6.5 shows the performance of *RElim* for all proposed methods: flat, *Full Expansion* (FE), *Progressive Expansion* (PE) and *Progressive Expansion k-by-k* (PEK). On the memory side (left plot), the effect of applying *PE* is clear. Note that the figure reports the total memory consumption, including both the first pass to compute 1-element item-sets and the rest of the mining process. With *PE*, the number of generated prefixes is dramatically reduced compared to *FE*, because of the

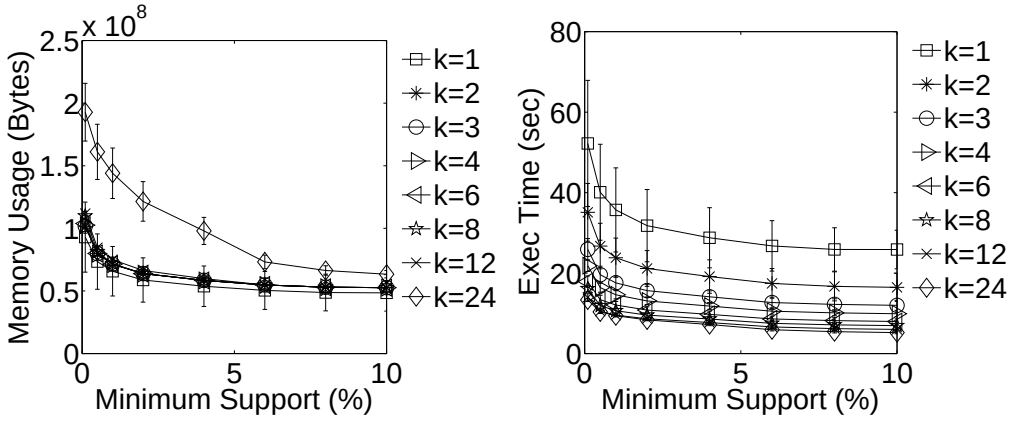


Fig. 6.6: Results for *Progressive Expansion k-by-k* on *dataset*. Memory usage (left) and execution time (right) for varying *minimum support* and different values of k.

fact that full IPs are rarely frequent enough to be above s and, therefore, they are generally cut before reaching /32. Consequently, the memory needed is far lower (e.g., $\approx 60\%$ reduction for the lowest *s* on *trace-1*). Moreover, as s goes up, *PE* gets closer to flat treatment. In Figure 6.2 (middle), we can also observe that the average transaction length decreases with s. In particular, for $s = 0.1\%$ the average length is ≈ 32 elements (32.23), while for $s = 10\%$ it hardly reaches 8 elements (8.06).

Nonetheless, the pruning task results in an important increase of the execution time. Figure 6.5 (right) shows that *PE* is slower than *FE*. This is due to the fact that *PE* must go through all transactions 25 times, each one for extending all prefixes by one bit until reaching /32 (in *FE* every transaction is read only once and each IP directly expanded into 25 items). The next section evaluates an optimization that addresses this issue.

Progressive Expansion k-by-k

Figure 6.6 plots the memory usage (left) and the execution time (right) for all possible values of k when using *Progressive Expansion k-by-k (PEK)*. In terms of memory, $k = 24$ is the worst option (it expands directly all trees and, therefore, needs a lot of memory), while all the other values of k behave much better. However, when switching to the runtime comparison, $k = 12$ and $k = 24$ turn out to be the best choices while $k = 1$ is the worst (it needs to analise all transactions 25 times). Clearly, $k = 12$ is the optimal choice.

Figure 6.5 shows that *PEK* with $k = 12$ outperforms both *FE* and *PE* in terms of

Table 6.3: Impact of sampling on *FaRNet* for *trace-1*: $mean \pm stdev$ for true positives rate (TPR) and false positives rate (FPR) for different *minimum supports (s)* and sampling rates (p).

	$p = 1\%$		$p = 10\%$		$p = 25\%$	
	TPR (%)	FPR (%)	TPR (%)	FPR (%)	TPR (%)	FPR (%)
$s = 0.1\%$	93.64 ± 0.32	1.38 ± 0.12	98.76 ± 0.16	0.9 ± 0.09	99.26 ± 0.15	0.55 ± 0.1
$s = 1\%$	97.26 ± 0.65	1.77 ± 0.45	98.87 ± 0.18	0.78 ± 0.22	99.33 ± 0.41	0.36 ± 0.09
$s = 2\%$	96.81 ± 1.62	1.24 ± 0.65	99.72 ± 0.27	0.12 ± 0.04	99.96 ± 0.03	0.11 ± 0.04
$s = 4\%$	95.76 ± 5.22	0.46 ± 0.36	98.33 ± 1.62	0.31 ± 0.28	99.36 ± 1.34	0.11 ± 0.21
$s = 6\%$	98.73 ± 1.2	2.47 ± 2.86	99.75 ± 0.26	1.46 ± 2.59	99.78 ± 0.26	0.23 ± 0.32
$s = 8\%$	98.94 ± 1.43	0.41 ± 0.76	100 ± 0	0 ± 0	100 ± 0	0 ± 0
$s = 10\%$	100 ± 0	1.32 ± 2.88	100 ± 0	1.29 ± 2.89	100 ± 0	0 ± 0

both execution time and memory consumption. Although *PEK* and *PE* use approximately the same memory, *PEK*'s execution time is vastly reduced with respect to *PE*'s (transactions are only read three times to generate /8, /20 and /32 prefixes).

Sampling

Next, we evaluate the impact of sampling on the output of *FaRNet*. We use the following metrics to quantify the error introduced by sampling. Let X be the union of flows matching all the frequent item-sets after running *FaRNet* on unsampled data. Likewise, Y defines this set of flows for the sampled case.

$$\text{True Positives Rate: } TPR = \frac{|X \cap Y|}{|X|}$$
$$\text{False Positives Rate: } FPR = \frac{|Y - X|}{|X|}$$

While *true positives* show the efficiency with respect to the unsampled case, the *false positives* indicate the correctness of the output (infrequent item-sets that turned frequent due to the sampling process).

In Table 6.3, we can observe the error introduced in the output of *FaRNet* due to sampling for varying s and sampling rate p on *trace-1*. Concerning the TPR, we obtain a percentage close to 94% in the worst case. Moreover, for $p > 1\%$, the TPR is always extremely close to 100% regardless of s. We see that higher p and s result in smaller error. As regards the FPR, the worst results are clearly obtained for the lowest p. The highest value is 2.47%. However, for higher p, the number of false positives is greatly reduced, especially for low s. The reason why smaller values of s lead to less false positives is because the number of frequent item-sets that turn out to be mistakenly reported as frequent, account for a low number of flows with respect to the overall input flows.

For example, for $s = 10\%$ and $p = 1\%$, we obtain $TPR = 100\%$ and $FPR = 6.42\%$. These false positives are due to the fact that in the unsampled output, there is an item-set whose frequency is very close to s (9.96%) that, after sampling, depending on the case, becomes frequent (*new item-set* phenomenon described in Section 6.2.4). This only mistakenly classified item-set is provoking that peak in the FPR.

All in all, we can conclude that for $p = 10\%$ and above, we obtain great accuracy and low false positives. Also, for the most aggressive sampling rate, $p = 1\%$, we get acceptable values of both TPR and FPR as long as we use reasonably high values of s.

In Figure 6.5, we can observe the memory usage and runtime when applying sampling to *FaRNet*. As expected, when we apply a non-aggressive $p = 25\%$, the improvement in the performance is significant. Runtime beats clearly *PEK* w/o sampling and becomes very close to flat treatment. Regarding memory usage, it becomes the best among all methods, including the flat case. With this sampling rate, *FaRNet* needs less than 4 seconds in the worst case to process a trace of 15 minutes with approximately half million flows (*trace-1*). This confirms that *FaRNet* performs (near) real-time.

Selecting the minimum support

Frequent item-sets are useful for many reasons (e.g., traffic profiling, anomaly extraction[23], anomaly detection [34]). They tell the operator what is happening in the network in a compact and summarised way. However, it is essential to tune the FIM system so that the amount of reported frequent item-sets is treatable by a human. While low s might lead to huge outputs with too much information, high s may hide interesting patterns. Therefore, finding the proper trade-off is crucial so that *FaRNet*'s output is useful for a network operator.

Figure 6.7 shows how the top-k of frequent item-sets varies for different s and data treatments (flat or hierarchical) when mining the 5-tuple on *trace-1* (see details in Table 6.2). While in the horizontal axis we see s from 0.1% to 10%, the vertical axis depicts the size of the output for a particular s (in logarithmic scale). We observe that for both flat and hierarchical attributes the number of item-sets decreases rapidly as s increases. However, the number of frequent item-sets changes significantly depending on how we treat the data. For flat attributes, the size of the output is always, at least, one order of magnitude lower than for hierarchical data, except for the most aggressive $s = 10\%$. This difference is due to the growth of frequent combinations between hierarchical elements, i.e., IP prefixes and ports. Therefore, in order to obtain a reasonable and humanly treatable number of item-sets in the output, the recommended s parameter changes significantly depending on the case.

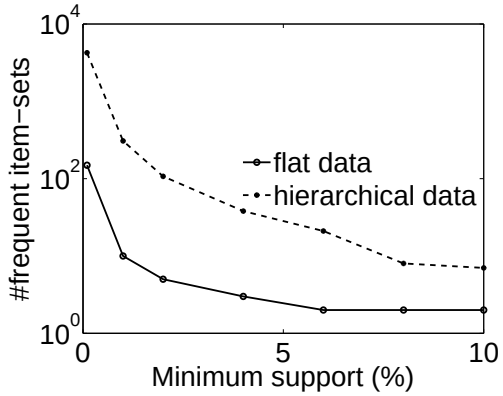


Fig. 6.7: Number of frequent item-sets for flat and hierarchical data and varying *minimum support* for *trace-1*.

Let us assume that we are interested in extracting the top-10 or top-20 of frequent item-sets. While we should use a $5\% \leq s \leq 10\%$ for hierarchical data, a s close to 1% would be more convenient for the flat case. For example, for *trace-1*, $s = 1\%$ returns 10 item-sets for flat treatment but, in order to reach a similar figure for the hierarchical case, s must be increased up to 8%.

6.3.3 Comparison with AutoFocus

AutoFocus [25] is the only available tool of similar nature to our system (see Chapter 6.5 for details on AutoFocus). In order to validate the implementation of *FaRNet*, we compare it against AutoFocus. For the comparison, we configured both systems so that the same threshold is used to decide whether an item-set is frequent. While *FaRNet* uses the *minimum support (s)*, AutoFocus uses a parameter called *resolution*. We also limited the number of dimensions analysed by *FaRNet* to only these attributes used by AutoFocus to perform the mining, i.e., five (the 5-tuple). For the experiments in this section, we used *trace-1* (see details in Table 6.2).

Performance comparison

Figure 6.8 shows how *FaRNet* and AutoFocus perform for different values of s. Note that only the first 10000 flows of *trace-1* are used for this comparison. This is because the available implementation of AutoFocus [29] is not dimensioned to handle more flows (when it receives more than that amount, it does not count them accurately due to collisions). In terms of runtime (right plot), *FaRNet* is clearly faster regardless of

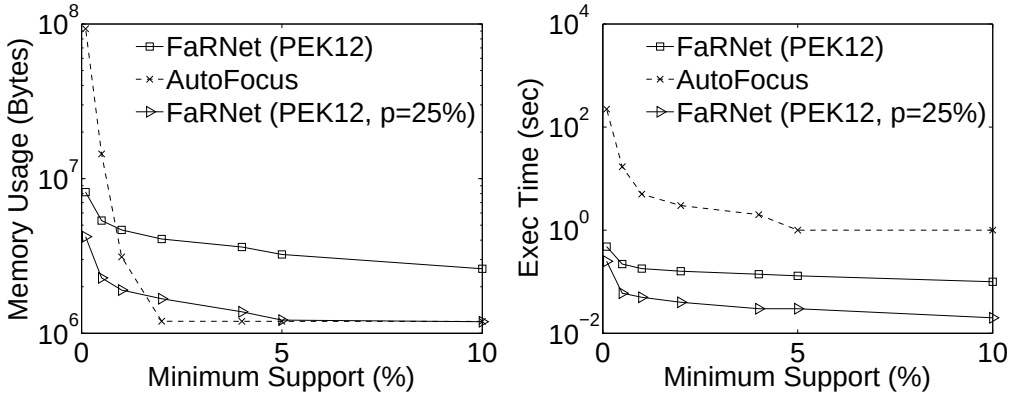


Fig. 6.8: Memory usage (left) and execution time (right) comparison between *FaRNet* (*PEK12*) and AutoFocus for the first 10K flows of *trace-1*.

s. Moreover, as s decreases, AutoFocus' execution time increases exponentially, while *FaRNet* is able to handle it smoothly. Although for the highest s AutoFocus' runtime ($1s$) is relatively close to *FaRNet*'s ($0.10s$), for $s = 0.1\%$ AutoFocus is approximately three orders of magnitude slower ($223s$ vs $0.48s$).

As regards the memory consumption (left plot), AutoFocus is better than *FaRNet* for $s \geq 1\%$. However, for lower values of s, AutoFocus consumption rises rapidly and ends up consuming far more memory than *FaRNet* (88.77 MB vs 7.79 MB for $s = 0.1\%$). All in all, *FaRNet* shows to be quicker and more resilient to low s than AutoFocus, although it uses more memory for $s = 1\%$ and above. However, note that the memory consumption of *FaRNet* (without sampling) is reasonably low in the worst case (below 10 MB).

In order to improve *FaRNet* results, we apply sampling to the input data. In particular, the triangle dotted line in Figure 6.8 shows *FaRNet* under a 25% sampling rate. As we can observe, *FaRNet* becomes faster than without sampling and, more importantly, gets really close to the memory consumption of AutoFocus. Under this sampling rate, *FaRNet*'s memory usage is lower than AutoFocus' for $s < 2\%$ and approximately the same for $s > 5\%$. However, for intermediate values of s (2% and 4%), *FaRNet* is still slightly worse.

Near real-time processing

An important advantage of *FaRNet* over AutoFocus is that *FaRNet* is capable of operating near real-time (i.e., it can process x minutes of data in less than x minutes). As shown in Figure 6.5, *FaRNet* without sampling can process 15 minutes of NetFlow

data (*trace-1*) in significantly less than 20 seconds regardless of the *minimum support*. Note that *FaRNet* works on fixed time windows, i.e., it collects data during a certain time bin and when it ends it processes that data and produces results in less than the time bin. While processing, *FaRNet* needs to store part of the next time bin in a buffer, otherwise some traffic would be lost.

When applying 25% sampling, *FaRNet* can analise *trace-1* with low error in less than 4 seconds in the worst case (lowest s). On the contrary, AutoFocus is not able to provide real-time results. As depicted in Figure 6.8, for the first 10,000 flows of *trace-1* (which correspond to approximately 18 seconds of traffic in our scenario), AutoFocus struggles for low s. For $s = 0.5\%$ it is slightly below the threshold of 18 seconds (16.25 seconds), but for $s = 0.1\%$, AutoFocus needs more than 12 times that time (almost 4 minutes).

Scalability to more dimensions

The main issue of AutoFocus is its lack of scalability as the number of dimensions increases. Essentially, AutoFocus combines several unidimensional hierarchies (trees) into a multi-dimensional bigger hierarchy. The problem of such a pass is that the combinations grow exponentially with more dimensions. This is because each new dimension considered must be "replicated" for every single node of every unidimensional hierarchy. In particular, the number of clusters above the threshold is bounded by $r \prod_{i=1}^{k} d_i$, where k is the number of dimensions considered, i.e., 5 when mining the 5-tuple (i.e., srcIP, dstIP, srcPort, dstPort and protocol). d is the depth of the hierarchy of dimension i (e.g., 25 for IPs) and r is the resolution [25]. Therefore, the more dimensions and the higher the depth of the hierarchies, the worse. In contrast, adding more dimensions to *FaRNet* is as easy as increasing the transaction length. We have already seen that *FaRNet* easily handles transaction lengths of 55 items for the 5-tuple. In next section, we report results on analysing the full set of 10 dimensions and show that *FaRNet* is able to process them very quickly.

6.3.4 Mining more dimensions

By mining the full set of 10 dimensions, item-sets like these reported in Table 6.4 can be obtained. In particular, this table shows the top-10 frequent item-sets reported by *FaRNet* on *trace-1* for s=1%. This top accounts for approximately 10% of the traffic. Note that a higher value of s could be used to summarise a higher percentage of the traffic in only 10 entries. The table also shows the item-sets returned by AutoFocus in

Table 6.4: Top-10 frequent item-sets reported by AutoFocus and *FaRNet* on *trace-1* sorted by descending frequency (*s=1%*). lp and hp stand for low-ports and high-ports, respectively.

	item-set	sIP	dIP	sPort	dPort	proto	app	sAS	dAS	sGeoloc	dGeoloc
AutoFocus	1	*	*	lp	hp	6	n/a	n/a	n/a	n/a	n/a
	2	*	*	80	hp	6	n/a	n/a	n/a	n/a	n/a
	3	*	*	hp	hp	6	n/a	n/a	n/a	n/a	n/a
	4	A.0.0.0/8	*	lp	hp	6	n/a	n/a	n/a	n/a	n/a
	5	*	*	hp	lp	6	n/a	n/a	n/a	n/a	n/a
	6	A.0.0.0/10	*	*	hp	6	n/a	n/a	n/a	n/a	n/a
	7	A.0.0.0/10	*	lp	hp	*	n/a	n/a	n/a	n/a	n/a
	8	A.0.0.0/9	*	80	hp	6	n/a	n/a	n/a	n/a	n/a
	9	A.14.29.110/32	*	80	hp	6	n/a	n/a	n/a	n/a	n/a
	10	*	*	hp	80	6	n/a	n/a	n/a	n/a	n/a
FaRNet	1	A.14.29.110/32	*	80	hp	6	HTTP	70	786	Bethesda	GB
	2	*	B.128.0.0/10	80	hp	6	HTTP	*	*	US	EU
	3	C.66.122.0/24	*	80	hp	6	HTTP	32	*	Stanford	EU
	4	C.66.120.0/21	*	80	hp	6	HTTP	32	786	Stanford	GB
	5	*	B.48.0.0/13	80	hp	6	HTTP	*	2200	US	FR
	6	D.234.254.12/30	*	80	hp	6	HTTP	6932	*	Danvers	EU
	7	E.100.0.0/18	*	hp	hp	6	*	1103	*	Amsterdam	*
	8	D.234.252.14/32	*	80	hp	6	HTTP	6932	*	Danvers	EU
	9	A.14.29.110/32	*	80	hp	6	HTTP	70	680	Bethesda	DE
	10	A.14.29.110/32	*	80	hp	6	HTTP	70	2200	Bethesda	FR

order to observe the differences among both systems. *FaRNet* is able to process this trace (≈ 0.5 million flows) in approximately 12 seconds.

As we can see in the table, while AutoFocus essentially reports quite general IP prefixes, *FaRNet* is able to find much richer item-sets containing more concrete prefixes, specific locations, applications and ASes. For example, the 9th item-set reported by AutoFocus describes the activity of a a host that, judging by the source port, could be a web server. However, other than its source IP, the remaining data of the item-set is not extremely useful. However, when looking at the output provided by *FaRNet*, we can see that item-sets 1, 9 and 10 uncover some very interesting hidden associations related to the very same item-set. In particular, we confirm that the host is indeed a web server (its associated application is HTTP), we find its source AS and also discover that it is located in a city called Bethesda (Maryland, US). Specifically, this web server hosts a widely used tool in the research community that gives access to a database of references on life science and biomedical topics. Moreover, we observe that it is being accessed from several European countries (Great Britain, Germany and France) and also find the corresponding destination AS for each case. Note that for the hierarchical attributes, only the most specific element is reported (e.g., in case of Bethesda, only the city is reported because adding the continent, the country and the region would be redundant). As we have been able to observe in this example, in order to understand what is really happening in the network, the item-sets reported by *FaRNet* are more synthetic and, therefore, much more informative than these returned by AutoFocus.

Table 6.5: List of item-sets found by *FaRNet* for a vertical scan detected by *NetReflex*.

sIP	dIP	sPort	dPort
X.191.64.165	Y.13.137.129	55548	*
Z.66.124.39	Y.13.137.129	*	*
*	Y.13.137.129	3072	80
*	Y.13.137.129	1024	80

6.4 Deployment

Two versions of *FaRNet* have been deployed in two different environments. The first deployment is described in this section and the second,

A prototype version of *FaRNet* [33] based on flat treatment on the 5-tuple [23] has been deployed in the European-wide backbone network of GÉANT (see scenario details in Chapter 2). It is essentially used by the network operators for automatically extracting and summarizing compactly all the traffic flows causing an anomaly (*anomaly extraction*). Even though *FaRNet* is a tool for traffic profiling, our main intuition for using it for *anomaly extraction* is that anomalies often result in large sets of flows with similar characteristics [23] (e.g., when there is a DDoS, lots of flows share the same destination IP and port). Consequently, anomalies will appear in the reports of *FaRNet* as frequent item-sets. In particular, *FaRNet* has been used for two purposes in GÉANT: 1) validate anomaly alarms previously triggered by an already existing anomaly detection tool in the network (*NetReflex* [54]) and 2) collect malicious evidence in a compact summary to send it to the involved parties.

We also deployed *FaRNet* in a different environment related with monitoring the behaviour of highly distributed devices in a critical infrastructure sector (refer to Appendix A for details).

The process of analysing an anomaly works as follows: 1) *NetReflex* triggers an alarm and provides the related meta-data (e.g., involved IPs), which is used to extract a set of candidate flows responsible for the anomaly; 2) each flow is modeled as an item-set with 5 items and *FaRNet* is used to extract the frequent item-sets out of the large set of candidate flows obtained in step 1. *FaRNet* also provides the network operators with a GUI that allows them to extract the frequent item-sets associated with an alarm, investigate the raw flows matching any returned item-set, and tune the extraction parameters if needed (e.g., the *minimum support*).

In order to ensure that the prototype of *FaRNet* was working as expected, we performed an evaluation during the deployment process in order to validate its results. During this evaluation, we randomly selected anomalies from 10 days. For 42% of the anomalies, *FaRNet* found uniquely the item-set strictly related to the meta-data

reported by *NetReflex*. Additionally, for 26% of the cases, the algorithm evidenced additional flows related to the anomaly that were not provided by the anomaly detector. These were particularly interesting cases because *FaRNet* was able to discover new anomalies that had been missed by *NetReflex*. For example, the following meta-data were signaled and labeled as a vertical scan by *NetReflex*: "sIP: X.191.64.165, dIP: Y.13.137.129, srcPort: 55548 and dstPort: *. When analysing the same anomaly using *FaRNet*, the frequent item-sets in Table 6.5 were found. The 1st was precisely the item-set responsible of the anomaly already flagged by *NetReflex*. The 2nd was another host doing a similar vertical scan on the same target, while the 3rd and 4th were two simultaneous DDoS on port 80 against the same target. We believe that this capability of finding more flows related to an anomaly has general applicability. Moreover, in 26% of the cases, some additional item-sets related to legitimate activity were extracted, which could be trivially filtered out by the network operator. For the remaining 6% of the alarms, *FaRNet* was not able to extract meaningful flows, which could be due to a stealthy anomaly not captured by our extraction technique or due to a false positive-alarm (FP).

Overall, *FaRNet* was extremely useful for the network operators of GÉANT. In particular, it provided the following:

1. Fast and more reliable anomaly analysis with respect to the time-consuming and error-prone manual investigation.

2. Discovery of additional information related to an anomaly that was missed by *NetReflex*.

3. Easier identification of FP reported by *NetReflex*.

4. Useful and compact summaries of the traffic.

6.5 Chapter Summary

We analysed the performance of state-of-the-art FIM algorithms when applied to network traffic data, and extended and optimized them to deal with hierarchical dimensions such as IP addresses, ports, applications and geolocation data. We also evaluated the impact of sampling on the performance of FIM algorithms, and showed that it significantly reduces computational overhead while providing precise output.

Based on this analysis, we built *FaRNet*, a network traffic profiling system that offers better performance and flexibility and also scales to a much higher number

of dimensions than AutoFocus. In order to validate the correctness of *FaRNet*, we compare it with AutoFocus by using a limited version of our system configured to produce the same output. Using traffic data from a large backbone network, we show that when mining only the 5-tuple, *FaRNet* is able to process 15 minutes of traffic in less than 4 seconds with a very small error. We show that *FaRNet* is up to three orders of magnitude quicker than AutoFocus. As a consequence, *FaRNet* is able to process high volumes of multi-dimensional traffic data in (near) real-time, while AutoFocus was designed for offline analysis of a pre-defined set of 5 dimensions. Finally, when analysing the full set of 10 dimensions, *FaRNet* confirmed its ability to produce more useful and synthetic reports and also showed that it scales very well with the number of dimensions by analysing 15 minutes of traffic in approximately 12 seconds (w/o sampling).

We deployed a preliminary version of *FaRNet* in the European backbone network of GÉANT and showed its usefulness and good results for assisting network engineers when dealing with anomalies in an operational environment.

The state-of-the art of this thesis is split into five sections: anomaly analysis (Section 6.6), anomaly detection (Section 6.7), extraction (Section 6.8) and classification (Section 6.9) and, finally, impact of sampling on anomaly detection (Section 6.10).

6.6 Anomaly Analysis

There are few studies providing feedback about traffic anomalies and its behaviour and characteristics in network traffic. However, none of these studies on his own can be considered representative enough because they are either based on short-term data (few weeks or months) or extracted from small networks (e.g., university links).

In [2], firewall logs were collected over four months from over 1600 different networks world wide (5 full class B networks, approximately 45 full Class C sized networks and many smaller sub-networks). This study revealed large intrusion activity on a daily basis and also found that a few sources were responsible for a significant part of the attacks. Moore et al. studied the broad visibility of flooding attacks and worms by using network telescopes [3, 4]. The unwanted traffic sent to network telescopes (range of monitored network addresses) was analysed and global activity for such attacks was inferred. [5] sought to characterize anomalous traffic by using 2 months of data during 2004. The analysed data was composed by one /8 network, two /19 networks and ten /24 subnets. It reported on temporal patterns and correlated activity among other characteristics. It concluded that the presence of worms and autorooters is significant. They provided as well some feedback on how to use their insights for monitoring and

detection purposes. More recently, [6] provided an analysis of the evolution of the scanning activity over 12.5 years (1994-2006). Network traffic was continuously collected at the border of the Lawrence Berkeley National Laboratory (LBNL) in Berkeley (USA). LBNL's address space consists of two /16 networks and a small number of /24 network blocks. It reports on the number of scanners as well as targets and probing patterns during that period of time.

Additionally, the closest studies we can find in the literature basically analysed general characteristics and evolution trends for the Internet traffic.

In [71], characteristics on wide area Internet from OC-3 (≈155 Mbps) links in commercial Internet backbone of MCI Communications Corp. [72] are presented. However, since the longest analysed trace lasts only fore one week, it can not be considered a long-term study. In [73], McCreary et al. reported results from the NASA Internet traffic exchange point. The study took into account 10 months of data, from May 1999 to March 2000 and showed that there was an increase of streaming and gaming traffic. In [74], a passive monitoring system called IPMON deployed in the Sprint backbone network [75] is described. In addition, changes in Internet traffic characteristics during 2001 and 2002 based on measurements from OC-3 (≈155 Mbps), OC-12 (≈622 Mbps) and OC-48 links(≈2.5 Gbps) are reported. In [76], a 5-year study (1998-2003) about traffic measurements from research, academic and commercial sites is presented. [77] reports a 7-year analysis (2001-2008) of long term behaviour of certain traffic characteristics.

6.7 Anomaly Detection

The amount of research on anomaly detection is huge and increasing every day. Furthermore, already existing threats change fast and new and unknown anomalies continuously show up. The state-of-the-art on anomaly detection can be divided in two general groups: specific-anomaly detection (Section 6.7.1) and general-purpose anomaly detection methods (Section 6.7.2). The first set includes concrete mechanisms to detect well-known anomaly types. The second describes techniques focusing on detecting anything that differs from usual, which means that mechanisms in this group can detect both known and unknown anomalies. Next, we review the most significant works. For more details please refer to [78].

6.7.1 Attack-specific Techniques

This Section covers the most relevant literature on the detection of these anomalies analysed in this thesis, i.e., scans (Section 6.7.1) and denial of service attacks (Section 6.7.1).

Scans

In [79], Lee et al. described the distinct scanning attacks and also defined their general behaviour. They differentiate three kind of scans: vertical, horizontal and a mixture between them. For details on scan types refer to Chapter 2.

There are lots of proposed solutions for scan detection. The most straightforward detection mechanism keeps a counter of the number of contacted destination ports and IPs from a given source IP for a fixed time window. This is implemented in the Snort IDS [9]. Nonetheless, this mechanism is not extremely effective as it totally depends on the time window, i.e., an attacker can launch a scanning attack at a very low rate to avoid being detection. Some works such as [80] focus particularly on detecting slow or stealthy scans, i.e., scanning attacks performed a at a very low rate. However the complexity of its computations renders this proposal unfeasible for high-speed links.

Consequently, more complex algorithms with higher detection rates have been proposed. Notably, Threshold Random Walk (TRW) [10] and Time based Access Pattern Sequential hypothesis testing (TAPS) [11]) are well-known techniques in the literature. TRW is implemented in the Bro IDS [12]. For details on TRW and TAPS refer to Chapter 2.

Several improvements have been proposed for TRW [81, 82, 83]. The main difference between original TRW and [81] is that the latter is able to catch scanners quicker. However, neither the original TRW nor this approach are able to scale to high-speed links because they do not take into account memory constraints (very limited memory access time and tiny memory size). Consequently, further improvements have been proposed for TRW in order to address these memory-related problems [82, 84, 85, 83, 86]. TAPS was also improved with TAPS-SYN [14], which showed a lower ratio of false positives with respect to the original TAPS. Finally, some mechanisms based on entropy measurements have also addressed the problem of scan detection [19, 87].

For further details refer to [83] and references therein.

DoS/DDoS

There is significant research dealing with DoS and DDoS attacks. In general, defense mechanisms against these attacks can be split into three general groups: 1) prevention,

2) detection, and 3) reaction. DoS prevention is the most effective defense mechanism because it aims to filter the attack before it actually happens. However, when a DoS occurs it is of crucial importance to be able to detect it in order to take the appropriate measures. Finally, attack reaction focuses essentially on reducing the effect of a DoS, i.e., trying to mitigate the attack and offer the expected service under attack.

One of the proposed solutions for DoS prevention focuses on filtering the traffic either entering or leaving a network by controlling if the incoming/outgoing packets are coming from or going to an expected range of IP addresses [88]. Similarly, Park and Lee proposed Router-based Packet filtering [89], which seeks to filter unexpected traffic inside a network rather than on the edges as in [88].

As regards DoS detection, many proposals have been made. In [90, 91], a mechanism that flags heavy changes in packet rates between hosts as DoS attacks was proposed. Statistical analysis was used for detection of SYN floods [92, 93]. The former [92] uses the ratio of SYN packets to FIN and RST packets while the latter [93] measures other metrics such as TCP traffic volume. In both cases, they build a profile for normal traffic and trigger a DoS when the observed traffic does not fit that profile. Spectral analysis is used in [94] to differentiate between normal and malicious flows by monitoring the amount of packets observed in a fixed time window. In [95], the authors exploit the traffic pattern observed when a DoS is happening, which differs clearly from legitimate traffic. Whereas a DDoS attack generates many traffic flows from different sources to the same dst and port, normal traffic is normally highly uncorrelated, i.e., many destination IPs and ports are visited. Such patterns are detected by using Kolmogorov-based algorithms. Time series analysis is employed to build a profile of normal traffic using e.g., the auto-regressive model and raises anomalies when the traffic differs from that model [96]. In [97], the authors propose a mechanism that monitors all the IPs interacting with the target. In general this set of IPs remains stable unless an attack occurs, in which case the system flags a DoS.

Signature-based approaches are not extremely useful to detect DoS attacks because DoS easily change its content and therefore designing accurate signatures is highly ineffective [84]. However, these sorts of techniques might be successful for detecting the communication between the attacker and its zombies (unless this interaction is encrypted, which happens often) [98]. In contrast, anomaly-based approaches are more appropriate as they are able to flag anything that differs from normal rather than depending on pre-defined anomalous patterns. In order to define what is normal, statistical definitions are used (e.g., IP packet length) [99, 100]. Afterwards, the similarity between the observed traffic and the normal profile must be determined in order to flag an attack. Forrest and Hofmeyr proposed LISYS, a system for Intrusion Detection that is able to detect an attack after some initial training to learn what

traffic is legitimate [101].

Reacting to an attack is crucial in order to allow legitimate users to use the attacked service as quickly as possible. The first step to reduce the impact of an attack is to protect the bottleneck, i.e., either the link or the end-host delivering the service [102]. Schuba et al. proposed a mechanism that is able to free allocated resources of suspicious TCP connections in the target (generated by a SYN flood) by sending RST packets [103]. The main idea in [104] is to ask legitimate clients of the service to increase their bandwidth usage to the maximum they can so that the traffic aggregate coming from them is higher than the malicious. If that is accomplished, legitimate hosts should be able to use their service. In [105], the authors propose booking resources for legitimate users, e.g., by using a server farm and a load balancer in order to improve the availability of the given service. Peng et al. present a scheme where the target host records these IPs that more often interact with it and when it is under attack it rejects any IP out of that history-based database [106]. An agent-cotroller mechanism is proposed in [107] for mitigating attacks within the domain of an ISP.

For more details on the state-of-the art on (D)DoS attacks, please refer to the available surveys [108, 109, 110].

6.7.2 General-purpose Techniques

The proposed solutions can be basically split into these general groups: statistical-based [111, 112, 113, 114], subspace-based [115, 116, 57, 19, 117, 118], signal-based [119, 120, 121] and behaviour-based [19, 87].

Statistical-based mechanisms use prediction to know what is the next expected value (time series prediction). If the measured metric differs too much from its prediction, it is considered to be an anomaly. Different periods of time are considered: weekdays, weekends, day, night, etc. For each combination of these intervals, two thresholds are defined: the upper one and the lower. When some specific measured metric (e.g., #pkts, #flows, #bytes...) exceeds its corresponding upper or lower value an anomaly is reported. There are several approaches. *Exponential smoothing* [111] is the simplest method for prediction: the predicted value is extracted from the average between the current prediction and the current real value. The problem of this approach is that it does not account for seasonality, so it would always report anomalies when there were fast changes due to normal activity. The other prediction model is called *The Holt-Winters Forecasting Algorithm* [111] and it tries to overcome the previous problem. Its prediction is an average of three variables that account for baseline, linear trend and seasonality respectively. Each of these variables is updated using exponential smoothing.

The subspace-method [57] analyses what the authors call OD-flows (flows with the same Origin and Destination points of the monitored network). Because of the high dimensional multivariate data structure of that flows, a lower-approximation is needed: Principal Component Analysis (PCA) [116]. This mathematical method captures the most important trends of the explored data (it preserves the significance of the data while reducing its complex initial structure). Then, the subspace method divides the resulting set into normal and anomalous. If the projection of the data in the second space is higher than a previously given value, an anomaly is flagged. All that procedure is known as (single-way) subspace method but there are other mechanisms based on that one like the multi-way subspace method [19]. There, Lakhina et al., focused on data distribution: they wanted to detect the anomalies according to changes on the distribution of some particular metrics. The way they studied the distribution of a certain data set is the entropy: the higher is the dispersion of the data the higher is the entropy value (a 0 value of entropy means no variation). They claim that entropy allows to detect unseen anomalies by volume metrics (e.g., number of packets or bytes). First, they compute the anomaly for each OD-flow and feature (sources and destinations IPs and ports) and then, they apply the before explained subspace method.

Next mechanisms are based on signal-processing techniques. In [119], wavelets are used. A wavelet transform is the procedure of dividing a given signal into different frequency components. Applied to anomaly detection it is used in such a way that each component is used to look for anomalies matching its scale: low-frequency components contain very sparsed anomalies, so they are used to detect long-time anomalies. High-frequency components are useful to detect spontaneous changes (short-term anomalies). In [120], Thottan et al. show the potential and challenges associated with applying signal processing techniques to the problem of network anomaly detection by presenting an mechanism based on abrupt change detection.

As regards behaviour-based mechanisms, in addition to [19], there are other different approaches like [87]. In that thesis, Xu et al, detect anomalies according to changes on the distribution of some specific measured metrics. The way they studied the distribution of a certain data set is the entropy: the more disperse the data is, the higher is its entropy value. First of all, for each host, the distribution probability is calculated for each metric: source IP, destination IP, source port and destination port. After that, the most significant values are selected from each set. For each of these values they observed how the other three related metrics changed (e.g., if testing a certain source IP, they investigated what happened with the destination IP, source port and destination port for all flows having that source IP). According to that three values they made their decision (e.g., a scanner IP would have a large accessed set of

destination IPs or ports with fixed or random source ports).

A recent research work [24] that does not fit in any of the before mentioned groups has been proposed. The main difference is that whereas all the others need to learn what normal traffic is, this approach skips that step and bases the detection only on observing changes in strongly correlated flows. For instance, in a DDoS, there will be lots of flows sharing the same destination IP and port, which will violate the expected equilibrium that is accomplished when traffic is normal. They claim to be more effective than other well-known approaches such as wavelets [119].

6.8 Anomaly Extraction

The usage of frequent item-set mining (FIM) for network monitoring and anomaly detection has been hardly treated as a research topic. In [23, 122], each flow is considered to be a transaction of length 7: source and destination IP, source and destination port, protocol, #packets and #bytes. Apriori algorithm is used to precisely identify all the traffic flows associated with an anomaly among a large set of candidate flows identified somehow by an anomaly detection system.

More recently, researchers have studied the related problem of finding hierarchical heavy hitters (HHH) [27, 28, 123, 124, 125]. Given a stream of items, e.g., IP addresses, a HHH is an aggregate, e.g., an IP address prefix, on a hierarchy that appears often. The HHH problem is, in fact, a special case of the more general FIM problem. HHH algorithms typically process the input data in a streaming fashion, approximate the HHHs, and can accommodate a small number of dimensions. Most FIM algorithms operate in an offline fashion, i.e., making multiple passes over input data, without approximation and scale better to a large number of dimensions. Furthermore, finding heavy hitters over streams of flat (network traffic) attributes is a widely-studied special case of the more general HHH problem (and in turn of the FIM problem). Heavy hitters are simply frequent items in an 1-dimensional stream of flat items.

AutoFocus [25] is a well-known system for finding HHHs over network traffic data. In contrast, to most other HHH algorithms, it operates in an offline manner making multiple passes over the input data. It takes as input 5-tuples of IP addresses, ports, and protocol, it treats IP addresses and ports as hierarchical attributes, and it finds frequent 5-dimensional aggregates. AutoFocus is the most related previous work. In fact, AutoFocus is essentially a particular case of *FaRNet*. The main contribution of our thesis is to build a new traffic profiling system based on FIM principles that is much faster, more flexible and more general than AutoFocus and allows to easily extend the input 5-tuples to include a much larger number of dimensions.

Later on, in [126], a more efficient version of Autofocus was proposed, although without making its code available. In [61], the subspace method (see Section 6.7.2) is combined with traffic sketches in order to provide both anomaly detection and later identification of those flows responsible for the anomaly. Recently, Silveira et al. proposed a technique called URCA [24] that iteratively reduces the candidate flows by removing those that seem to be normal according to the feedback received by the anomaly detector. Note that it does not depend on a particular anomaly detection system but only on the type of feedback it receives from it, and, therefore, it is not tied to a particular detection approach. After isolating the anomalous flows, it clusters together similar anomalies.

6.9 Anomaly Classification

Only few works have strictly addressed the problem of anomaly classification. By anomaly classification we refer to identifying precisely the high-level anomaly type (e.g., network scan) of an alarm rather than only reporting the alarm (i.e., not giving only a merely binary response normal/anomalous traffic). Note that obtaining labeled data, i.e., annotating anomalous traffic with representative and precise anomaly classes is extremely challenging because of both its intrinsic complexity and the highly dynamic nature of anomalous traffic.

Some proposals have presented techniques to inject synthetic anomalies in a dataset in order to obtain a labeled training dataset for supervised anomaly detection [127, 128]. Other works assume the availability of training datasets, which is very difficult as it would be necessary to have all the anomalies appearing in a dataset properly characterized [129, 130]. Lakhina et al. [19] cluster the output of a PCA-based anomaly detector to identify anomalies with similar behaviour. Nonetheless, human intervention is still necessary to find the correspondence between each reported cluster and the high-level anomaly that it is describing. Tellenbach et al. [20] classify changes to generalized entropy metrics of traffic feature distributions to identify the type of detected anomalies. They show how this approach can classify synthetic anomalies with an accuracy of $\approx 85\%$. Finally, Choi et al. [21] make use of parallel coordinate plots to find unique patterns of attacks that are easy to recognize visually by a human expert.

6.10 Impact of Sampling on Scan Detection

There is not a lot of research dealing with the impact of sampling on scan detection but some work had been already done: [13, 53, 14, 15, 16, 17]. All these works, essentially analyse the performance of existing scan detection mechanisms by testing them under different sampling techniques. More recently, some researchers have also directed their efforts towards improving that performance rather than only analysing it [121, 131].

Mai et al. studied the impact of packet sampling on portscan detection in [14]. TRW, TAPS and an entropy-based anomaly detection technique were tested. Concerning TRW, they found out that flow size became lower in the presence of sampling, thus resulting in more false positives and a significant increase of the success ratio. They also showed that the metric used by TAPS (relation between number of accessed destination IPs and ports) is less affected, thus concluding that TAPS is better than TRW under sampling. When comparing both mechanisms under sampling, they observed that, while TRW had better success ratio, TAPS exhibited a lower ratio of false positives. Regarding to the entropy-based analysis, it presented a performance very similar to TAPS mechanism due to the fact that both mechanisms detect scanners profiling destination access patterns. They even proposed a new portscan detection method based on TAPS called TAPS-SYN, that reduced the false positives ratio.

In [13], the same authors tested several sampling methods (*Packet Sampling*, *Flow sampling*, *Sample-and-Hold* and *Smart Sampling*) against two portscan detection mechanisms (TRW and TAPS) and a wavelet-based volume anomaly detection mechanism. They found out that *Flow Sampling* performed better for both portscan and volume anomalies detection, while the other methods obtained very poor results. While *Packet Sampling* suffered the well-known flow shortening, *Sample and Hold* and *Smart Sampling* showed to be not suitable for anomaly detection since they look down on small flows (and attacks like port scans and DoS are made using that kind of flows).

In [53], Brauckhoff et al. studied how specific metrics are affected by sampling looking at counts of bytes, packets and flows, together with several feature entropy metrics. They concluded that entropy summarisation is more resilient to sampling than volume-based metrics.

In [15], the impact of sampling on a Change-Point Detection (CPD) method [113] and a PCA-based anomaly detection mechanism [132], is analysed. They made the experiments under several sampling methods proposed in the PSAMP IETF draft [133]: *Systematic sampling*, *Random n-out-of-N Sampling* and *Uniform Probabilistic Sampling*. They concluded that *Sistematic Sampling* is almost useless for anomaly detection when the anomaly detection mechanism relies on specific packet characteristics (e.g., TCP flags). When flow-based metrics are used instead, the performance is

directly related to the sampling rate. Regarding the PCA-based method, the same behaviour is observed (it does not care about the particular sampling method used).

In [17], Androulidakis et al. designed a new flow-based sampling technique called *Selective Sampling* that focuses on small flows. These flows are usually responsible of port scanning and DoS activity. Then, they tested how it worked under a CPD anomaly detection mechanism. Moreover, its performance is compared with another flow-based sampling method (*Smart Sampling*) and against *Random Flow Sampling*. This new sampling technique showed to improve the anomaly detection effectiveness and, sometimes, it even outperformed the unsampled case. Unlike in previous works [14], they observed that *Smart Sampling* overcame *Flow Sampling* in some cases, thus concluding that it can be used for anomaly detection if the used detection mechanism relies on large flows.

In [16], the authors extended their previous work testing their new sampling technique against a PCA-based anomaly detection mechanism. They observed that *Selective Sampling* exposed the anomalies more clearly than *Flow Sampling* did, and, in some cases, even better than the unsampled case. They attributed this improvement to the fact that the nine metrics that they used for PCA were more correlated in small-flows (*Selective Sampling* focuses on small flows).

Recently, two works [121, 131] have focused on providing solutions to the impact of sampling rather than analysing the impact itself as previous studies did. In [121], Brauckhoff et al. propose an approach based on signal-processing that seems to improve significantly the performance of anomaly detection techniques under packet sampling. In [131], an algorithm called Progressive Security-Aware Packet Sampling is proposed in order to sample traffic in such a way that you keep mainly malicious part. Consequently, the performance achieved by the anomaly detector behind it is higher than when it receives randomly sampled data.

Chapter 7

Conclusions

This thesis has addressed some of the current practical challenges for anomaly detection in backbone networks by providing updated and detailed information on current anomalies, analysing and improving the impact of sampling and building a system for anomaly detection, extraction and classification. Furthermore, we also showed how the presented system can be successfully generalised to perform fast recognition of high-dimensional patterns from big network traffic data and successfully deployed it in two operational environments.

First, we analysed three commercial tools for anomaly detection and provided a study about the type and characteristics of the current security threats happening in a large backbone network. We also reported the strengths and shortcomings found while using these tools, as well as the experience and knowledge we acquired during this long process.

Second, we studied the impact of sampling on two scan detection algorithms to determine if they are robust enough to continue finding port scans reliably. In contrast to previous works, we observed that when using the same fraction of packets as the common metric to compare, *Packet Sampling* performed better than *Flow Sampling* for scan detection. Moreover, we showed that *Selective Sampling*, a recent sampling proposal that targets small flows, achieves the best overall performance for scan detection. The main problem of this technique is that it needs to keep all flows in memory in order to determine their size. This is costly in terms of memory and does not work online. Accordingly, we proposed a new implementation called *Online Selective Sampling* that works on a per-packet basis and keeps the same good results reported by *Selective Sampling*. Due to the ability to drop flows when they become large, our sampling proposal uses significantly less resources and is implementable in the widely deployed packet-based NetFlow.

Third, we presented a novel scheme to detect, extract and classify network anomalies that combines frequent item-set mining with machine learning. We showed that our solution has a very high classification accuracy (above 98%), and a low false positive rate (approximately 1%). Moreover, we were able to see that the model performed remarkably well in a network where it was not trained, which means that there are certain features that are independent from the environment.

Finally, we generalised our system and showed that it can be successfully used for extracting knowledge out of high multi-dimensional network traffic data, which can greatly help network operators to comprehend what is shaping their traffic. Additionally, a prototype of our system was deployed in two different scenarios to further show its usefulness. First, it was used in GÉANT to assist engineers in the investigation of network anomalies and second, it helped to monitor the malfunctioning of highly distributed devices related with a critical infrastructure sector.

In summary, the key contributions of this thesis are the following:

- It provides a long-term analysis on current network anomalies happening in a backbone network, which lacks in the literature as access to data from large networks is uncommon for the research community. This feedback will help researchers to focus on the most problematic anomalies and achieve a better understanding of their behaviour and characteristics.

- It reports on the practical limitations of current commercial tools for anomaly detection, which are not usually accessible for researchers (they are, in general, too expensive). This information will allow the research community to focus on still to be improved real world problems.

- Unlike previously reported, it shows that *Packet Sampling* outperforms *Flow Sampling* under the same fraction of packets, which is important as most routers only support packet-based sampling.

- It proposes *Online Selective Sampling*, a packet-based implementation of *Selective Sampling*, that is able to keep the same good accuracy for scan detection while using much less resources.

- It shows that frequent item-sets summarise all the traffic flows of an anomaly very well, which allows a better characterization of network attacks and thus, the creation of more accurate models for anomaly classification using machine learning.

- It shows that techniques based on frequent item-set mining are faster, more flexible and more scalable to high-dimensional data than methods based on hierarchical heavy hitters (HHH). Furthermore, it also shows that, by mining more dimensions, the traffic summaries produced are much more synthetic and useful than those limited to e.g., the 5-tuple produced by HHH-based algorithms like AutoFocus.

7.1 Future Work

In this thesis we have only covered few of the many challenges for anomaly detection faced by network operators. Moreover, the problems we tackled allow further research and improvements. For example, it is essential to work with up-to-date data on current anomalies in order to point research towards the right direction. Therefore, although we provided a long-term analysis on the behaviour of nowadays anomalies, it is necessary that this kind of data is updated from time to time due to the continuously changing network environment.

As regards the impact of sampling on anomaly detection, we have focused on a particular type of attack using two specific detection schemes under four sampling techniques. Therefore it is plenty of opportunities for future research either by analysing another sort of anomaly, another detection technique or by using other sampling techniques. Furthermore, as the speed of network links increases, the need for more aggressive sampling rates will be mandatory and investigating the performance of state-of-the art algorithms under these new conditions will be important as well.

As we explained during the manuscript, both the system we proposed for automatic anomaly detection, extraction and classification and *FaRNet* work near real-time, i.e., on fixed time-windows. However, data is neither static nor finite. On the contrary, data streams are infinite and dynamic. Consequently, several assumptions are not valid any more if these systems have to work real-time. For instance, data can be read only once and item-sets can change from frequent to infrequent over time.

Bibliography

[1] K.-K. R. Choo, "The cyber threat landscape: Challenges and future research directions," *Computers & Security*, vol. 30, no. 8, pp. 719–731, Nov. 2011.

[2] V. Yegneswaran, P. Barford, and J. Ullrich, "Internet intrusions: Global characteristics and prevalence," vol. 31, no. 1, pp. 138–147, Sept. 2003.

[3] D. Moore, G. Voelker, and S. Savage, "Inferring internet denial-of-service activity," in *Proc. of USENIX Security Symp.*, Aug 2001.

[4] D. Moore, C. Shannon, G. Voelker, and S. Savage, "Network telescopes: Technical report," *CAIDA*, Apr. 2004.

[5] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson, "Characteristics of internet background radiation," in *Proc. of ACM SIGCOMM Conf. on Internet Measurement (IMC)*, Oct. 2004.

[6] M. Allman, V. Paxson, and J. Terrell, "A brief history of scanning," in *Proc. of ACM SIGCOMM Conf. on Internet Measurement (IMC)*, Oct. 2007.

[7] Cisco Systems, "Cisco IOS NetFlow," http://www.cisco.com/en/US/products/ps6601/prod_white_papers_list.html.

[8] M. Molina, I. Paredes-Oliva, W. Routly, and P. Barlet-Ros, "Operational experiences with anomaly detection," *Computers & Security*, vol. 31, no. 3, pp. 273 – 285, May 2012.

[9] M. Roesch, "Snort–lightweight intrusion detection for networks," in *Proc. of USENIX Systems Administration Conf. (LISA)*, Nov. 1999.

[10] J. Jung, V. Paxson, A. Berger, and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," *Proc. of IEEE Symposium on Security and Privacy*, May 2004.

[11] S. Avinash, T. Ye, and B. Supratik, "Connectionless portscan detection on the backbone," *Proc. of IEEE Intl. Performance Computing and Communications Conf.*, Apr. 2006.

[12] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Computer networks*, vol. 31, no. 23-24, pp. 2435–2463, Dec. 1999.

[13] J. Mai, C. Chuah, A. Sridharan, T. Ye, and H. Zang, "Is sampled data sufficient for anomaly detection?" in *Proc. of ACM SIGCOMM Conf. on Internet Measurement (IMC)*, Oct. 2006.

[14] J. Mai, A. Sridharan, C. Chuah, H. Zang, and T. Ye, "Impact of packet sampling on portscan detection," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 24, no. 12, pp. 2285–2298, Dec. 2006.

[15] G. Androulidakis, V. Chatzigiannakis, S. Papavassiliou, M. Grammatikou, and V. Maglaris, "Understanding and evaluating the impact of sampling on anomaly detection techniques," in *Proc. of IEEE Military Communications Conf. (MILCOM)*, Oct. 2006.

[16] G. Androulidakis, V. Chatzigiannakis, and S. Papavassiliou, "Using selective sampling for the support of scalable and efficient network anomaly detection," in *Proc. of IEEE Globecom Workshops*, Nov. 2007.

[17] G. Androulidakis and S. Papavassiliou, "Intelligent flow-based sampling for effective network anomaly detection," in *Proc. of IEEE GLOBECOM*, Nov. 2007.

[18] G. Androulidakis, V. Chatzigiannakis, and S. Papavassiliou, "Network anomaly detection and classification via opportunistic sampling," *IEEE Network*, vol. 23, no. 1, pp. 6–12, Feb. 2009.

[19] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *Proc. of ACM SIGCOMM*, Aug. 2005.

[20] B. Tellenbach, M. Burkhart, D. Schatzmann, D. Gugelmann, and D. Sornette, "Accurate network anomaly classification with generalized entropy metrics," *Computer Networks*, vol. 55, no. 15, pp. 3485–3502, Oct. 2011.

[21] H. Choi, H. Lee, and H. Kim, "Fast detection and visualization of network attacks on parallel coordinates," *Computers & Security*, vol. 28, no. 5, pp. 276–288, Jul. 2009.

[22] M. Molina, W. Routly, I. Paredes-Oliva, and A. Jain, "Anomaly detection in backbone networks: Building a security service upon an innovative tool." in *Proc. of Terena Networking Conf. (TNC)*, May 2010.

[23] D. Brauckhoff, X. Dimitropoulos, A. Wagner, and K. Salamatian, "Anomaly extraction in backbone networks using association rules," in *Proc. of ACM SIGCOMM Conf. on Internet Measurement (IMC)*, Nov. 2009.

[24] F. Silveira and C. Diot, "Urca: Pulling out anomalies by their root causes," in *Proc. of IEEE INFOCOM*, Mar. 2010.

[25] C. Estan, S. Savage, and G. Varghese, "Automatically inferring patterns of resource consumption in network traffic," in *Proc. of ACM SIGCOMM*, Aug. 2003.

[26] Cisco Systems, "Sampled NetFlow," http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/12s_sanf.html.

[27] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava, "Finding hierarchical heavy hitters in data streams," in *Proc. of Intl. Conf. on Very Large Data Bases (VLDB)*, Sept. 2003.

[28] ——, "Diamond in the rough: Finding hierarchical heavy hitters in multi-dimensional data," in *Proc. of ACM SIGMOD*, Jun. 2004.

[29] AutoFocus implementation, http://www.caida.org/tools/.

[30] I. Paredes-Oliva, P. Barlet-Ros, and J. Solé-Pareta, "Portscan detection with sampled netflow," in *Proc. of Intl. Workshop on Traffic Monitoring and Analysis (TMA)*, May 2009.

[31] ——, "Scan detection under sampling: A new perspective," *IEEE Computer Magazine*, vol. 46, no. 4, pp. 38–44, Apr. 2013.

[32] I. Paredes-Oliva, P. Barlet-Ros, and M. Molina, "Automatic validation and evidence collection of security related network anomalies," in *Proc. of Passive and Active Measurement Conf. (PAM) (poster)*, Apr. 2010.

[33] I. Paredes-Oliva, X. Dimitropoulos, M. Molina, P. Barlet-Ros, and D. Brauckhoff, "Automating root-cause analysis of network anomalies using frequent itemset mining," in *Proc. of ACM SIGCOMM (demo)*, Sept. 2010.

[34] I. Paredes-Oliva, I. Castell-Uroz, P. Barlet-Ros, X. Dimitropoulos, and J. Solé-Pareta, "Practical Anomaly Detection based on Classifying Frequent Traffic Patterns," in *Proc. of IEEE Global Internet Symp. (GI)*, Mar. 2012.

[35] I. Paredes-Oliva, P. Barlet-Ros, and X. Dimitropoulos, "FaRNet: Fast Recognition of High Multi-Dimensional Network Traffic Patterns," in *Proc. of ACM SIGMETRICS (extended abstract)*, Jun. 2013.

[36] Cisco, "Cisco," http://www.cisco.com/.

[37] Sampling and Filtering Techniques for IP Packet Selection (RFC 5475), Mar. 2009, http://www.ietf.org/rfc/rfc5475.txt.

[38] N. Duffield, "Sampling for passive internet measurement: A review," *Statistical Science*, vol. 19, no. 3, pp. 472–498, Aug. 2004.

[39] N. Duffield and C. Lund, "Predicting resource usage and estimation accuracy in an ip flow measurement collection infrastructure," in *Proc. of ACM SIGCOMM Conf. on Internet measurement (IMC)*, Oct. 2003.

[40] M. J. Zaki and C.-J. Hsiao, "Charm: An efficient algorithm for closed itemset mining," in *2nd SIAM international conference on data mining*, vol. 15, 2002, pp. 457–73.

[41] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," *ACM SIGMOD Record*, vol. Vol. 22, pp. 207–216, Jun. 1993.

[42] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *ACM SIGMOD Record*, vol. Vol. 29, pp. 1–12, Jun. 2000.

[43] M. Zaki, S. Parthasarathy, M. Ogihara, and W. L. et al., "New algorithms for fast discovery of association rules," in *Proc. of Intl. Conf. on Knowledge Discovery and Data Mining*, Aug. 1997.

[44] C. Borgelt, "Keeping things simple: finding frequent item sets by recursive elimination," in *Proc. of Workshop on Open Source Data Mining Software (OSDM)*, Aug. 2005.

[45] C. Borgelt and X. Wang, "Sam: A split and merge algorithm for fuzzy frequent item set mining," in *Proc. of EUSFLAT Conf.*, Jul. 2009.

[46] J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent pattern mining: current status and future directions," *Data Mining and Knowledge Discovery*, vol. 15, no. 1, pp. 55–86, Aug. 2007.

[47] C. Borgelt, "Simple algorithms for frequent item set mining," *Advances in Machine Learning II*, vol. 263, pp. 351–369, Dec. 2009.

[48] DANTE, http://www.dante.net.

[49] GÉANT, http://www.geant.net.

[50] Anella Científica, http://www.cesca.cat/en/communications/anella-cientifica.

[51] RedIRIS, http://www.rediris.es/index.php.en.

[52] Centre de Supercomputació de Catalunya (CESCA), http://www.cesca.cat/en.

[53] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina, "Impact of packet sampling on anomaly detection metrics," in *Proc. of ACM SIGCOMM Conf. on Internet Measurement (IMC)*, Oct. 2006.

[54] Guavus, "NetReflex," http://www.guavus.com.

[55] Arbor Networks, "PeakFlow SP," http://www.arbornetworks.com/.

[56] Lancope, "StealthWatch," http://www.lancope.com/.

[57] A. Lakhina, M. Crovella, and C. Diot, "Characterization of network-wide anomalies in traffic flows," in *Proc. of ACM SIGCOMM Conf. on Internet Measurement (IMC)*, Oct. 2004.

[58] IANA port numbers, http://www.iana.org/assignments/port-numbers.

[59] Josep Sanjuàs-Cuxart, "Efficient Algorithms for Passive Network Measurement," Ph.D. dissertation, Universitat Politècnica de Catalunya BarcelonaTech (UPC), Mar. 2012.

[60] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, Jul. 1970.

[61] X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina, "Detection and identification of network anomalies using sketch subspaces," *Proc. of ACM SIGCOMM Conf. on Internet Measurement (IMC)*, Oct. 2006.

[62] G. Grahne and J. Zhu, "Efficiently using prefix-trees in mining frequent itemsets," in *Proc. of Workshop on Frequent Itemset Mining Implementations (FIMI)*, Dec. 2003.

[63] B. Goethals and M. Zaki, "Fimi'03: Workshop on frequent itemset mining implementations," in *Proc. of Workshop on Frequent Itemset Mining Implementations (FIMI)*, Nov. 2003.

[64] G. Grahne and J. Zhu, "High performance mining of maximal frequent itemsets," May 2003.

[65] P. E. Proctor, "Marketscope for network behavior analysis," *Gartner Research Report G00144385*, Nov. 2006.

[66] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: Myths, caveats, and the best practices," in *Proc. of ACM CoNEXT*, Dec. 2008.

[67] J. Quinlan, *C4. 5: Programs for Machine Learning*, Oct. 1992.

[68] P. Barlet-Ros, J. Solé-Pareta, J. Barrantes, E. Codina, and J. Domingo-Pascual, "Smartxac: A passive monitoring and analysis system for high-speed networks." *Campus-Wide Information Systems*, vol. 23, no. 4, pp. 283–296, Dec. 2006.

[69] V. Carela-Español, P. Barlet-Ros, A. Cabellos-Aparicio, and J. Solé-Pareta, "Analysis of the impact of sampling on netflow traffic classification," *Computer Networks*, vol. 55, no. 5, pp. 1083–1099, Abr. 2011.

[70] Christian Borgelt's Software, http://www.borgelt.net/software.html.

[71] K. Thompson, G. Miller, and R. Wilder, "Wide-area internet traffic patterns and characteristics," *IEEE Network*, vol. 11, no. 6, pp. 10–23, 1997.

[72] MCI Communications Corporation, http://www.mci.com/.

[73] S. McCreary and K. Claffy, "Trends in wide area ip traffic patterns," Citeseer, Tech. Rep., 2000.

[74] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S. Diot, "Packet-level traffic measurements from the sprint ip backbone," *IEEE Network*, vol. 17, no. 6, pp. 6–16, 2003.

[75] Sprint, www.sprint.com.

[76] M. Fomenkov, K. Keys, D. Moore, and K. Claffy, "Longitudinal study of internet traffic in 1998-2003," in *Proc. of the winter international synposium on Information and communication technologies*, 2004, pp. 1–6.

[77] 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*, 2009, pp. 711–719.

[78] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, Jul. 2009.

[79] C. Lee, C. Roedel, and E. Silenok, "Detection and characterization of port scan attacks," 2003.

[80] S. Staniford, J. A. Hoagland, and J. M. McAlerney, "Practical automated detection of stealthy portscans," *Journal of Computer Security*, vol. 10, no. 1, pp. 105–136, Jan. 2002.

[81] S. E. Schechter, J. Jung, and A. W. Berger, "Fast detection of scanning worm infections," in *Proc. of Recent Advances in Intrusion Detection (RAID)*, Sep. 2004.

[82] N. Weaver, S. Staniford, and V. Paxson, "Very fast containment of scanning worms," in *Proc. of USENIX Security Symp.*, Aug. 2004.

[83] S. Nam, H. Kim, and H. Kim, "Detector sherlock: Enhancing trw with bloom filters under memory and performance constraints," *Computer Networks*, vol. 52, no. 8, pp. 1545–1566, Jun. 2008.

[84] R. R. Kompella, S. Singh, and G. Varghese, "On scalable attack detection in the network," in *Proc. of ACM SIGCOMM Conf. on Internet Measurement (IMC)*, Oct. 2004.

[85] Q. Zhao, A. Kumar, and J. Xu, "Joint data streaming and sampling techniques for detection of super sources and destinations," in *Proc. of ACM SIGCOMM Conf. on Internet Measurement (IMC)*, Nov. 2005.

[86] J. Mikians, P. Barlet-Ros, J. Sanjuas-Cuxart, and J. Solé-Pareta, "A practical approach to portscan detection in very high-speed links," in *Proc. of Passive and Active Measurement (PAM)*, Mar. 2011.

[87] K. Xu, Z. Zhang, and S. Bhattacharyya, "Profiling internet backbone traffic: behavior models and applications," in *Proc. of ACM SIGCOMM*, Aug. 2005.

[88] P. Ferguson, "Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing (rfc 2827)," May 2000, http://tools.ietf.org/html/rfc2827.html.

[89] K. Park and H. Lee, "On the effectiveness of route-based packet filtering for distributed dos attack prevention in power-law internets," in *Proc. of ACM SIGCOMM Conf.*, Aug. 2001.

[90] T. M. Gil and M. Poletto, "Multops: a data-structure for bandwidth attack detection," in *Proc. of USENIX Security Symp.*, Aug. 2001.

[91] S. Abdelsayed, D. Glimsholt, C. Leckie, S. Ryan, and S. Shami, "An efficient filter for denial-of-service bandwidth attacks," in *Proc. of IEEE GLOBECOM*, Dec. 2003.

[92] H. Wang, D. Zhang, and K. G. Shin, "Detecting syn flooding attacks," in *Proc. of IEEE INFOCOM*, Jun. 2002.

[93] R. B. Blazek, H. Kim, B. Rozovskii, and A. Tartakovsky, "A novel approach to detection of denial-of-service attacks via adaptive sequential and batch-sequential change-point detection methods," in *Proc. of IEEE Systems, Man and Cybernetics*, Jun. 2001.

[94] C.-M. Cheng, H. Kung, and K.-S. Tan, "Use of spectral analysis in defense against dos attacks," in *Proc. of IEEE GLOBECOM*, Nov. 2002.

[95] A. Kulkarni, S. Bush, and C. Evans, "Detecting distributed denial-of-service attacks using kolmogorov complexity metrics," Dec. 2001, technical Report 2001CRD176, GE Research & Development Center.

[96] J. B. Cabrera, L. Lewis, X. Qin, W. Lee, R. K. Prasanth, B. Ravichandran, and R. K. Mehra, "Proactive detection of distributed denial of service attacks using mib traffic variables-a feasibility study," in *Proc. of IEEE/IFIP Intl. Symp. on Integrated Network Management*, May 2001.

[97] T. Peng, C. Leckie, and K. Ramamohanarao, "Proactively detecting distributed denial of service attacks using source ip address monitoring," in *Proc. of IFIP-TC6 Networking*, May 2004.

[98] G. Cheng, "Malware faq: Analysis on ddos tool stacheldraht v1.666," 2006, http://www.sans.org/security-resources/malwarefaq/stacheldraht.php.

[99] Z. Zhang, J. Li, C. Manikopoulos, J. Jorgenson, and J. Ucles, "Hide: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification," in *Proc. of IEEE Workshop on Information Assurance and Security*, Jun. 2001.

[100] C. Manikopoulos and S. Papavassiliou, "Network intrusion and fault detection: a statistical anomaly approach," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 76–82, Oct. 2002.

[101] S. A. Hofmeyr and S. Forrest, "Architecture for an artificial immune system," *Evolutionary computation*, vol. 8, no. 4, pp. 443–473, 2000.

[102] J. K. Millen, "A resource allocation model for denial of service," in *Proc. of IEEE Symp. on Security and Privacy*, May 1992.

[103] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram, and D. Zamboni, "Analysis of a denial of service attack on tcp," in *Proc. of IEEE Symp. on Security and Privacy*, May 1997.

[104] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker, "DDoS defense by offense," in *Proc. of ACM SIGCOMM*, Sept. 2006.

[105] F. Kargl, J. Maier, and M. Weber, "Protecting web servers from distributed denial of service attacks," in *Proc. of Intl. Conf. on World Wide Web*, May 2001.

[106] T. Peng, C. Leckie, and K. Ramamohanarao, "Prevention from distributed denial of service attacks using history-based ip filtering," in *Proc. of IEEE ICC*, Aug. 2003.

[107] U. K. Tupakula and V. Varadharajan, "A practical method to counteract denial of service attacks," in *Proc. of Australian Computer Science Conf.*, Feb. 2003.

[108] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 34, no. 2, pp. 39–53, Apr. 2004.

[109] C. Douligeris and A. Mitrokotsa, "Ddos attacks and defense mechanisms: classification and state-of-the-art," *Computer Networks*, vol. 44, no. 5, pp. 643–666, Oct. 2004.

[110] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the dos and ddos problems," *ACM Computing Surveys (CSUR)*, vol. 39, no. 1, p. 3, Apr. 2007.

[111] P. Brockwell, R. Davis, and I. NetLibrary, "Introduction to time series and forecasting," Mar. 2002.

[112] J. Brutag, "Aberrant behavior detection and control in time series for network monitoring," in *Proc. of USENIX Systems Administration Conf. (LISA)*, Dec. 2000.

[113] H. Wang, D. Zhang, and K. Shin, "Change-point monitoring for the detection of dos attacks," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, vol. 1, no. 4, pp. 193–208, Dec. 2004.

[114] A. Soule, K. Salamatian, and N. Taft, "Combining filtering and statistical methods for anomaly detection," in *Proc. of ACM SIGCOMM Conf. on Internet Measurement (IMC)*, Oct. 2005.

[115] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *Proc. of ACM SIGCOMM*, 2004.

[116] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. Kolaczyk, and N. Taft, "Structural analysis of network traffic flows," *Proc. of the joint international Conf. on Measurement and modeling of computer systems*, pp. 61–72, 2004.

[117] H. Ringberg, A. Soule, J. Rexford, and C. Diot, "Sensitivity of pca for traffic anomaly detection," *ACM SIGMETRICS Performance Evaluation Review (PER)*, vol. 35, pp. 109–120, Jun. 2007.

[118] D. Brauckhoff, K. Salamatian, and M. May, "Applying pca for traffic anomaly detection: Problems and solutions," Apr. 2009.

[119] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proc. of ACM SIGCOMM Workshop on Internet Measurment (IMW)*, Nov. 2002.

[120] M. Thottan and C. Ji, "Anomaly detection in ip networks," *IEEE Transactions on Signal Processing*, vol. 51, pp. 2191–2204, Aug. 2003.

[121] D. Brauckhoff, K. Salamatian, and M. May, "A signal processing view on packet sampling and anomaly detection," in *Proc. of IEEE INFOCOM*, Mar. 2010.

[122] D. Brauckhoff, X. Dimitropoulos, A. Wagner, and K. Salamatian, "Anomaly extraction in backbone networks using association rules," ETH Zurich, TIK-Report 309, Sept. 2009.

[123] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava, "Finding hierarchical heavy hitters in streaming data," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 4, pp. 2:1–2:48, Jan. 2008.

[124] J. Hershberger, N. Shrivastava, S. Suri, and C. Tóth, "Space complexity of hierarchical heavy hitters in multi-dimensional data streams," in *Proc. of ACM SIGMOD/PODS*, Jun. 2005.

[125] Y. Zhang, S. Singh, S. Sen, N. Duffield, and C. Lund, "Online identification of hierarchical heavy hitters: algorithms, evaluation, and applications," in *Proc. of ACM SIGCOMM Conf. on Internet Measurement (IMC)*, Oct. 2004.

[126] J. Wang, D. Miller, and G. Kesidis, "Efficient mining of the multidimensional traffic cluster hierarchy for digesting, visualization, and anomaly identification," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 24, no. 10, pp. 1929–1941, Oct. 2006.

[127] I. Steinwart, D. Hush, and C. Scovel, "A classification framework for anomaly detection," *Journal of Machine Learning Research*, vol. 6, no. 1, p. 211, Dec. 2006.

[128] N. Abe, B. Zadrozny, and J. Langford, "Outlier detection by active learning," in *Proc. of ACM SIGKDD*, Aug. 2006.

[129] D. Dasgupta and N. Majumdar, "Anomaly detection in multidimensional data using negative selection algorithm," in *Proc. of Congress on Evolutionary Computation (CEC)*, Jun. 2002.

[130] D. Dasgupta and F. Nino, "A comparison of negative and positive selection algorithms in novel pattern detection," in *Proc. of IEEE Systems, Man and Cybernetics*, Jun. 2000.

[131] S. Ali, I. Haq, S. Rizvi, N. Rasheed, U. Sarfraz, S. Khayam, and F. Mirza, "On mitigating sampling-induced accuracy loss in traffic anomaly detection systems," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 3, pp. 4–16, Jul. 2010.

[132] V. Chatzigiannakis, S. Papavassiliou, G. Androulidakis, and B. Maglaris, "On the realization of a generalized data fusion and network anomaly detection framework," in *Proc. of Intl. Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, Jul. 2006.

[133] IETF Packet Sampling (PSAMP) Working Group, http://www.ietf.org/html.charters/psamp-charter.html.

[134] Barcelona Digital Technology Centre, www.bdigital.org/en/.

Appendix A

Real-world Use Case

A.1 Introduction

This appendix shows how the methodology used in chapters 5 and 6 can be successfully applied in a different environment. In particular, it shows how frequent item-set mining (FIM), in conjunction with other techniques, can be successfully used to find and Analise anomalies in an extensive network of highly distributed electronic devices related with a critical infrastructure sector.

This appendix of the thesis is part of a collaboration with a private company (*Barcelona Digital Technology Centre* [134]), and, therefore, specific details on the scenario as well as on the data used and the monitored devices cannot be disclosed. This appendix fits in a bigger project that seeks to create a solution to predict and automate the process of detecting and mitigating anomalous situations related to the availability, security, and usability of such devices. In this appendix we report on the very first steps of this project.

In an extensive network of devices with lots of data ranging from status updates to malfunctioning reports or activity summaries, FIM is a particularly interesting method for mining data efficiently and discovering frequent patterns and strong correlations. For example, it can be used to find the most problematic devices of the network. Also, due to its capability of finding frequent correlations, it might be able to, for example, find devices failing simultaneously. This would allow investigating (also with data mining) any potential common root-cause problems producing such malfunctioning. Finally, this appendix also focuses on applying time series analysis for predicting the behaviour of the devices, which is extremely helpful for optimizing resources in, e.g., logistics. It would be possible to foresee the number of operatives needed to fix a

Table A.1: Datasets details.

Label	Init Date	End date	Duration	#records
devices-info	-	-	-	12000
daily-agg	Nov.'10	Oct.'12	2 years	5.83×10^6
breakdowns	Jan.'12	Jan.'13	1 year	98.48×10^3
daily-summ	Jan.'12	Nov.'12	316 days	635.56×10^6
operational	Jan.'12	Jan.'13	1 year	151.13×10^6

certain number of devices that, with high probability, will not work properly.

The rest of this appendix is organized as follows. First, Section A.2 describes the datasets used and the followed methodology. Next, Section A.3 reports on the results obtained in this phase of the project and finally, Section A.4 summarises the conclusions we extracted after using our methodology in a different environment.

A.2 Scenario and Methodology

A.2.1 Datasets

The monitored devices are distributed all over Spain. These devices generate lots of data, ranging from daily activity summaries to specific reports on their failures. Specifically, the available data sources are described below. For details on the duration and the magnitude of each dataset, please refer to Table A.1.

- Devices info: static information of each device (e.g., date it entered the network or model).
- Location: where is the device placed (e.g., zip code, city and region).
- Daily aggregations: per-day reports that specify several percentages indicating how a device worked during that day for a particular metric (e.g., 95% availability).
- Breakdown summaries: this report shows exact times when a device failed and started working again.
- Daily summaries: for each device, it shows its work flow. In particular, for each component it has, it is possible to monitor if it worked properly or if it had any malfunctioning and for how long.

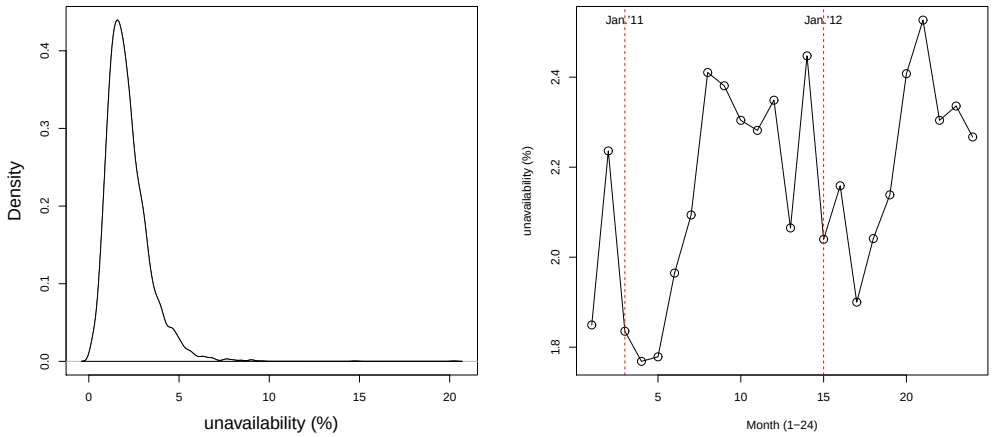


Fig. A.1: Unavailability distribution (left) and its evolution in the last two years (right).

- Operational data: it summarises the activities performed by a particular device (e.g., number of operations and type).

A.2.2 Methodology

The first step was to integrate all data sources to create a single per-device profile that contained all its associated information. Afterwards, we proceeded with a preliminary analysis in order to understand the new environment. In particular, we focused on analysing the unavailability of the devices, which is the most critical resource. We studied its distribution, its temporal evolution and its value depending on their location among others. Once we were familiarized with the new domain, we focused on solving the following two problems: 1) finding the most problematic devices and 2) predicting its unavailability for specific time periods. Finally, we centered our efforts towards the visualization of these results, which is also of foremost importance to provide timely results that facilitate a quick response.

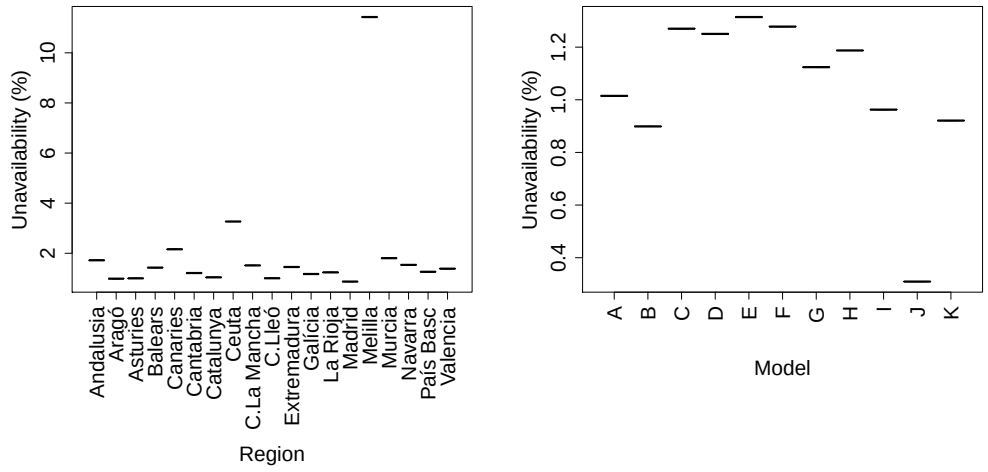


Fig. A.2: Unavailability depending on the location (left) and the model of the device (right).

A.3 Proposed solution

A.3.1 Preliminary analysis

Figure A.1 shows the distribution of the average unavailability of the devices (left). As we can observe, it follows a normal distribution with a slightly positive skew, which means that the mass of the distribution is more concentrated on the left of the figure (lower percentages of unavailability). The mean is 2.16% and the median 1.94%. The Q3 quartile indicates that 75% of the devices have an average unavailability below 2.72%. The 95th percentile is 4.38% and the maximum value observed is 20.23%.

The right plot of Figure A.1 presents the evolution of the unavailability for the two years of the historical (*daily-agg*). Two main conclusions can be extracted: 1) the availability of the devices has worsened in 2012 with respect to 2011 and 2) there are significant seasonal patterns. In particular, we can see that the first months of each year (January to March) are the best and that December and summer (June and July) are the worst.

Figure A.2 analyses the unavailability depending on the location (left) and the device model (right). As we can observe, the location seems to play a significant role in the availability of a device. Specifically, the are five zones where the value is notably higher (Melilla, Ceuta, Canàries, Andalusia i Múrica). One factor that might explain such differences is the extreme temperatures affecting these zones particularly. Regarding the model, we can also observe that there are important differences. Note

that for privacy reasons, real model names are hidden. Model J presents an availability that is far better than the rest. Moreover, we can observe important differences among the other models.

A.3.2 System architecture

For obvious reasons, we cannot disclose the real architecture of the proposed solution. Nonetheless, we describe how the reported results fit inside the proposed solution. The algorithms used in this block will be part of an "alert subsystem" that will be fed by many data sources (Section A.2.1) and will generate smart and proactive alerts (sections A.3.3 and A.3.4). Afterwards, this alerts will be sent to the visualization subsystem (Section A.3.5), among others, which are out of the scope of this thesis.

A.3.3 Problematic devices

In this section, we present the results obtained when using Frequent Item-Set Mining (FIM) for analysing which are the most problematic devices. In this environment, we used FIM as follows. First, we defined as problematic all devices with an unavailability above 5% on a particular day. Thus, the first step to calculate the devices impacting the most on the network, we compute a list of problematic devices for every day in the *daily-agg* dataset. In FIM terminology (Chapter 2), each of these lists is a transaction. Using FIM, we are able to quickly compute the most problematic devices (devices failing a percentage of days above the specified *minimum support s*). We perform this analysis for the whole network. However, in order to obtain actionable information, we also analyse the devices causing more problems per-zone so that identifying the worst behaving devices per-area can be used intelligently when organizing the operatives sent to fix them.

Likewise Chapter 5, the solution proposed in this appendix also uses FIM for mining flat data. In contrast, note that Chapter 6 presented a system that uses FIM for dealing with hierarchical elements. However, while in the present appendix and Chapter 6 transactions have a variable number of elements (e.g., list of devices that failed for a particular day in this case), in Chapter 5 they are always composed by 5 items (5-tuple). Regarding the algorithms used, while Chapter 5 used FPmax* and Chapter 6 used Apriori (flat data) and RElim (hierarchical data), in this appendix we used Eclat. As explained in Chapter 6, the reason for selecting a FIM algorithm depends on the dataset characteristics (e.g., transaction length). As we can observe in Figure A.3, in this case we are dealing with much longer transactions (167.98 items per transaction on average), which changes significantly the scenario. Moreover, a significant amount

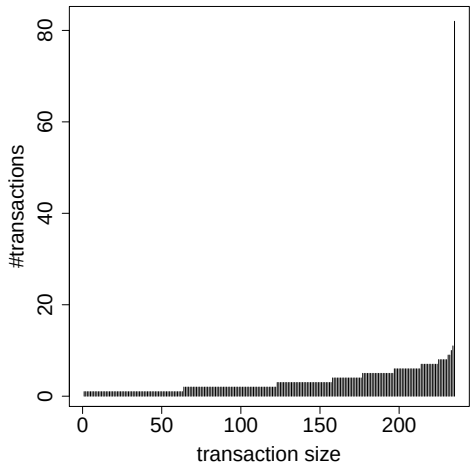


Fig. A.3: Daily transaction size distribution of problematic devices for all the network.

of transactions reach the maximum transaction length, which is far higher than the mean (232).

As an additional requirement, in this case, rather than focusing on the best tradeoff between memory and runtime as in e.g., Chapter 6, the algorithm giving the quickest response was preferred. However, for completeness with respect to previous chapters, this appendix also reports on the memory consumption. Figure A.4 shows the performance (mean$\pm$stdev) of the evaluated FIM algorithms i.e., Apriori, Eclat, FP-growth, RElim and SAM. On the left plot, we observe the memory used for values of s ranging from 1% to 10%. Apriori turns out to be, by far, the worst performing algorithm as already reported (Chapter 6). Regarding the remaining techniques, they did not present any significant differences. In contrast, when looking at the execution time of each algorithm (Figure A.4, right plot), the differences are noticeable among each other. Overall, after analysing both the runtime and the memory used for the mining by each of the algorithms, *Eclat* turns out to be the best algorithm.

Using s=10%, FIM reports that the top-5 problematic devices of the network fail 27.06%, 22.18%, 18.17%, 17.17% and 16.04% of the days, respectively. All reported frequent item-sets are of size one, i.e., one single device. Nonetheless, lowering *s* to 5%, FIM is able to find longer frequent item-sets, i.e., with more than one device, which means that they malfunctioned simultaneously. This is very interesting as it gives the opportunity to further investigate for possible common root-cause analysis affecting both devices by applying FIM on other data describing these devices. However, this

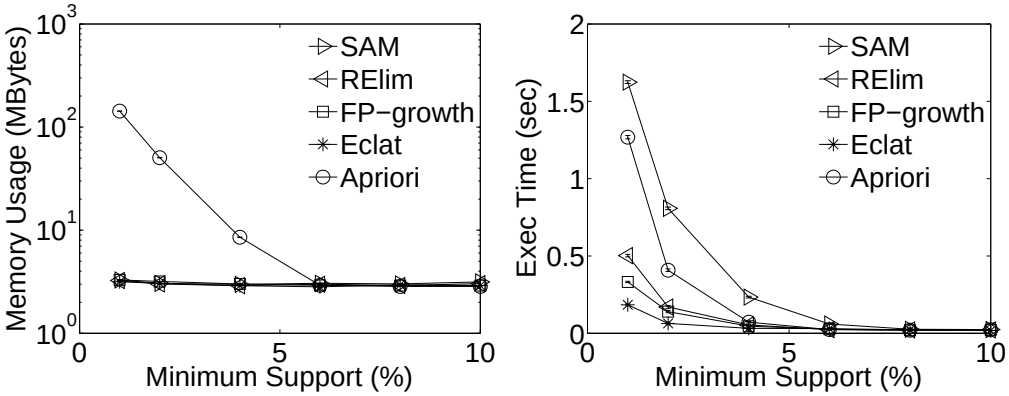


Fig. A.4: Comparison among FIM algorithms tested.

has not yet been addressed in the project.

Regarding the most problematic devices for each zone, the main difference resides on the smaller size of the transactions as the number of daily problematic devices is much lower for a single zone (with respect to the whole network). Specifically, on average, the number of problematic devices is ≈ 1.032, which holds extremely similar performance results in terms of both runtime and memory usage for all the FIM algorithms.

A.3.4 Behaviour prediction

In order to organize resources properly, it is very important to be proactive rather than reactive to problems. Even though operatives will have to go to the place where a device is malfunctioning, if you are able to know that it will fail in advance, it is possible to e.g., efficiently design the followed route or correctly dimension the number of operatives needed depending on the number of problems predicted. In order to address this issue we used time series analysis. Specifically, we employed Holt Winters [111] and ARIMA models (see Chapter 6.5 for details). We designed models for specific devices and also for zones.

Figure A.5 shows an example of the prediction for the unavailability of specific device. The black line stands for the historical data while the blue represents the prediction. The colored areas around the prediction are the 80% (dark blue) and the 95% (light blue) confidence intervals respectively. In the first case, we realized that making daily predictions for each device was not possible as the behaviour did not show any clear daily pattern. However, when aggregating the data weekly, the models were

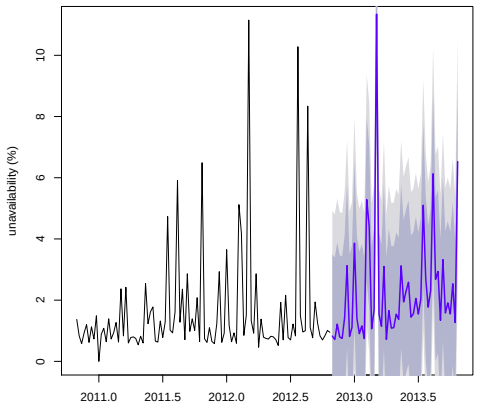


Fig. A.5: Weekly prediction of the unavailability of a single device for the next 6 months.

able to detect some patterns as we can see in the figure.

Figure A.6 shows the per-zone prediction for the number of devices that will be down. Any device is considered to be down when its availability is above 5%. Note that in contrast to the prediction presented in Figure A.5, we are forecasting the absolute number of devices that will have problems rather than their unavailability percentage individually. Since a zone includes many devices, the model is able to catch both daily and weekly patterns (left and right plots, respectively).

In order to determine the goodness of a created model, we used the root-mean-square error (RMSE), which measures the error between the values predicted by the model and the values actually observed. Therefore, the lower the RMSE, the better the model is. This measure is particularly useful because it uses the same scale of the predicted variable, i.e., number of devices that will be down in the former example. This is interesting as it allows to have a quick idea of whether the error of the model is acceptable or not. The RMSE can only be used for comparing models predicting the same variable.

A.3.5 Visualization

Representing data visually is a particularly effective method to understand it quickly. Visualization is especially useful if it shows the information in the geographical location where it is actually occurring. For instance, if there is a certain prediction of some

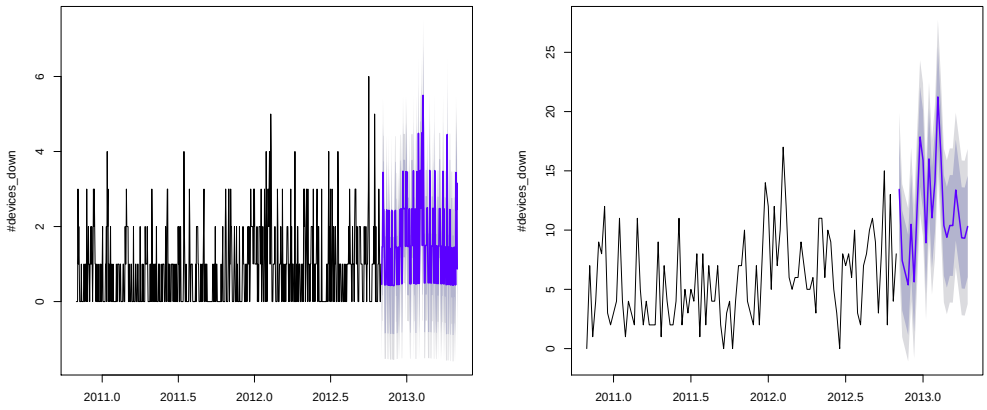


Fig. A.6: Daily (left) and weekly (right) prediction of the number of devices that will be down for a particular zone.

malfunctioning devices for a city for the next week, it is straightforward to spot where there will be more problems by plotting and observing circles of different sizes or colors in each zone of the city. Another example could be monitoring the whole network of devices in real-time in order to identify failures or any other metric in a rapid manner.

Indeed, when we represented e.g., the predicted unavailability in a map, we realized about some clear patterns we had not seen before. We saw that while in specific zones of a city the problems concentrated on the weekends, for other areas, the problems were higher for other days. However, we can not give further details on this. Instead, Figure A.7 shows the physical distribution of the devices, which gives an idea of the potential of this type of representation. While the black points represent devices, the shadowed areas indicate how dense a specific area is. In particular, the color of the shadows goes from dark blue (low density) to light blue (high concentration of devices).

A.4 Chapter Summary

In this appendix we have presented a real-world deployment that successfully uses our methodology in a different environment. In particular, we have showed the usefulness of FIM for mining large amounts of data and finding interesting correlations quickly in an extensive network of highly distributed devices. Furthermore, we reported how

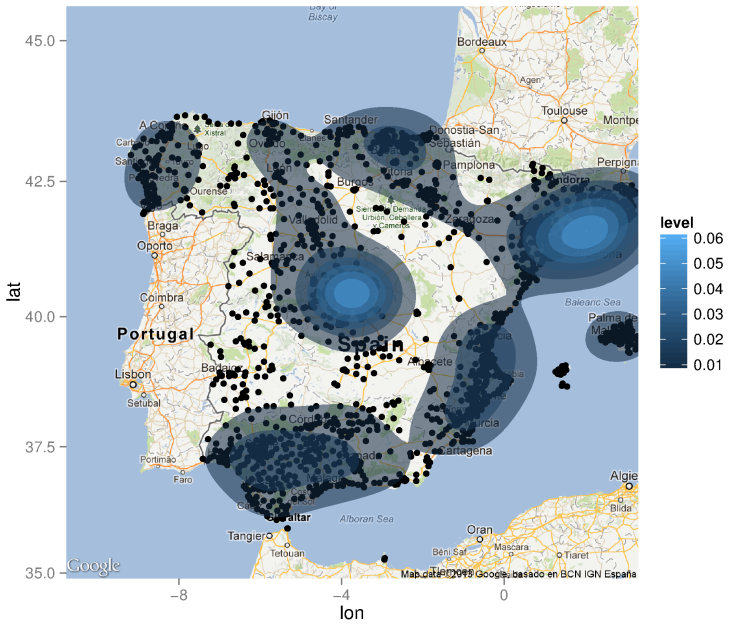


Fig. A.7: Devices location in the map.

time series analysis serves the purpose of predicting the behaviour of these devices.

The number of devices present in the network has grown significantly during the last couple of years and the tendency to keep rising seems quite clear. Therefore, the ability to deal with an ever increasing amount of data is a matter of growing necessity. As a consequence, using efficient algorithms for analysing big data and creating actionable knowledge will be a must-do for business.

Appendix B

Publications

B.1 Journals

- (Under Review) **I. Paredes-Oliva**, P. Barlet-Ros and X. Dimitropoulos. FaRNet: Fast Recognition of High-Dimensional Patterns from Big Network Traffic Data. Computer Networks, 2013.

- **I. Paredes-Oliva**, P. Barlet-Ros and J. Solé-Pareta. Scan Detection under Sampling: A New Perspective. IEEE Computer Magazine, Special Issue on Cybersecurity, vol. 46, no. 4, pp. 38-44, 2013.

- M. Molina, **I. Paredes-Oliva**, W. Routly and P. Barlet-Ros. Operational Experiences with Anomaly Detection in Bakbone Networks. Computers & Security, vol. 31, no. 3, pp. 273-285, 2012.

B.2 Conferences

- **I. Paredes-Oliva**, P. Barlet-Ros and X. Dimitropoulos. FaRNet: Fast Recognition of High Multi-Dimensional Network Traffic Patterns. In Proceedings of ACM SIGMETRICS (extended abstract and poster), Pittsburgh, PA, USA, 2013.

- **I. Paredes-Oliva**, I. Castell-Uroz, P. Barlet-Ros, X. Dimitropoulos and J. Solé-Pareta. Practical Anomaly Detection based on Classifying Frequent Traffic Patterns. In Proceedings of IEEE Global Internet Symposium (GI), Orlando, FL, USA, 2012.

- **I. Paredes-Oliva**, X. Dimitropoulos, M. Molina, P. Barlet-Ros and D. Brauckhoff. Automating Root Cause Analysis of Network Anomalies using Frequent Itemset Mining. In Proceedings of ACM SIGCOMM Conference (extended abstract and demo), New Delhi, India, 2010.

- M. Molina, W. Routly, **I. Paredes-Oliva** and A. Jain. Anomaly Detection in Backbone Networks: Building a Security Service Upon an Innovative Tool. In Terena Networking Conference (TNC), Vilnius, Lithuania, 2010.

- **I. Paredes-Oliva**, P. Barlet-Ros and M. Molina. Automatic Validation and Evidence Collection of Security Related Network Anomalies. In Passive and Active Measurement Conference (PAM) (extended abstract and poster), Zurich, Switzerland, 2010.

- **I. Paredes-Oliva**, P. Barlet-Ros and J. Solé-Pareta. Portscan Detection with Sampled NetFlow. In Proceedings of International Workshop on Traffic Monitoring and Analysis (TMA), Aachen, Germany, 2009.

B.3 Technical Reports

- **I. Paredes-Oliva**, P. Barlet-Ros and J. Solé-Pareta. Analysis of the Impact of Traffic Sampling on Portscan Detection. Technical Report ref. UPC-DAC-RR-CBA-2009-14, 2009.