


Doctoral Dissertation
Doctoral Program in Electrical, Electronics and Communications Engineering
(34^{th} cycle)

An AI and data-driven approach to unwanted network traffic inspection

Francesca Soro

* * * * * *

Supervisors
Prof. Marco Mellia, Supervisor
Prof. Antonio Lioy, Co-supervisor
Prof. Idilio Drago, Co-supervisor

Doctoral Examination Committee:
Prof. Matthias Wählisch, Referee, Freie Universität Berlin
Prof. Battista Biggio, Referee, University of Cagliari
Prof. Gareth Tyson, Queen Mary University of London
Prof. Oliver Hohlfeld, Brandenburg University of Technology
Prof. Cataldo Basile, Politecnico di Torino

Politecnico di Torino
January 11^{th}, 2022

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

. . . . . .

Francesca Soro

Turin, January 11th, 2022

Summary

The growing number of connected devices on the Internet makes the end users more and more vulnerable to cyberattacks. Malicious entities in the network are constantly fostering the rise of new botnets and crafting new threats that, if successful, may directly impact critical infrastructures and people's everyday life. Detecting these attacks in real-time is paramount to properly counteract them, but it is also complex, due to the volume, variety and velocity of the data travelling on the Internet. In this scenario, collecting and analyzing unwanted darknet traffic - often referred to as Internet Background Radiation – may be a path to take to detect new and potentially malicious phenomena. Observing the traffic hitting these fully passive probes, a network analyst can identify for instance heavy-hitter sources, service requests to vulnerable target or new coordinated events. Moreover, coupling the fully passive probes together with active honeypots further enriches the visibility on malicious events. In contrast to the darknets, honeypots are indeed able to reply to unwanted requests, providing a broader knowledge on the threat scenario. A manual inspection of these traffic traces is however impossible. For this reason, in this thesis I present a framework to automatically extract knowledge from unwanted traffic data captured on darknets and honeypots. Given the characteristics of the dataset at my disposal, I make extensive use of big-data and machine learning techniques to reach my goal. I first provide a characterization of the traffic hitting fully passive probes located in three different parts of the world, highlighting the type of traffic they receive, the most targeted services and their differences and similarities. As a second step, I enrich my scenario with an active honeypot infrastructure, capable of replying to service requests with different levels of complexity. I demonstrate that actively engaging with the senders increases the volume of traffic, and more complex responses push the attackers to reach further attack stages. After a thorough characterization of the most relevant network events, I proceed to the data analytics phase: I first depict the activity on the network as a graph, on top of which I test different community detection algorithms to group together similar activity patterns. By means of such techniques I am able to distinguish between communities devoted to vertical and horizontal scans, for instance, or recognize some more fine-grained patterns. As a final step, I benchmark a well-known set of anomaly detection algorithms against a novel AI

technique, providing a set of custom metrics to quantify their detection capabilities, even when no ground truth is available. My results demonstrate how the adoption of big-data and machine learning techniques ease the network traffic monitoring and analysis task, highlighting potentially critical events that would otherwise go unnoticed.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The widespread adoption of Information and Communication Technologies has had an ever growing impact on everyday life, profoundly changing the way in which people all around the world communicate, work and access any kind of service and content. According to the Cisco Annual Internet Report [37], the number of internet users around the world is expected to grow up to 5.3 billion by 2023, with an average of 3.6 networked devices per capita. The categories of connected devices are more and more variate as well, spanning from classical laptops and mobile phones, to IoT and industrial control devices.

Together with the number of connected devices, the number of possible threats and attacks towards them is constantly rising, calling for new techniques to protect the end users. Real-time detection of cyber-threats is paramount to properly counteract these events, but it is also a complicated task, requiring an high level of automation. Indeed, the extremely high volume of traffic reaching the communication infrastructures everyday, as well as its velocity and variety, makes a manual traffic inspection impossible.

The inspection of unwanted traffic - also referred to as Internet Background Radiation (IBR) - hitting passive targets in the network may help to detect the rise of new threats. In this scenario, darknets, i.e., completely passive targets, are some of the most commonly investigated sources of information. Darknets are sets of IP addresses which expose no services and do not answer to any request, and should therefore observe no activity towards them, nor receive service requests. In practice, instead, many remote sources reach for them for the most variate purposes, from malicious scans in search of vulnerable targets, to benign actors testing the IP space for research purposes.

Characterizing the traffic hitting the darknet may already help in detecting scanning attempts, both malicious or related to research projects, or heavy-hitter sources, but gives little to no information on more sophisticated operations. To further characterize the behavior and intentions of senders, active targets as the honeypots are needed. Honeypots mimic specific set of services at different levels.

They are left vulnerable on purpose, in order to lure attackers and capture as much information as possible about their behavior, including the set of commands they used, their login attempts or malware source code, as well as zero-day attacks.

In this thesis I describe the deployment of a comprehensive framework for the analysis of unwanted traffic, bringing together darknets and different types of honeypots. I test the feasibility and usefulness of such framework in a cybersecurity scenario defining i) which kind of phenomena are detectable and ii) which are the steps required to identify them. Through the chapters I describe how I captured traffic and how I extracted knowledge from the analysis of raw packets. Given the significant volume of traffic hitting such destinations everyday, *big-data* and *machine learning* techniques are necessary. Most of the chapters are taken from papers published in international conferences and journals. At the beginning of each chapter, I report the venue in which the content has been presented.

In Chapter 2, I first summarize the state of the art in darknet and honeypot traffic analysis, collecting the most relevant works addressing this topic.

Chapter 3 illustrates the main characteristics of the datasets I use throughout the thesis. I describe the full knowledge extraction pipeline, from the raw data capture to the log aggregation and the data analytics phase. I capture traffic hitting 3 darknet deployments in different geographic location starting from 2018, and update the infrastructure with traffic hitting an active honeypot deployment starting from end 2020. Finally, I discuss the ethical implication of the honeypot activity and data collection.

The first work presented in this thesis is contained in Chapter 4. This chapter provides a first characterization of darknet traffic, comparing three deployment of different sizes and geographical locations. I investigate to what extent the events on the network do depend on the addresses location, on the observation time and on the darknet size, and what the most evident changes across deployments are. I drill down on the traffic categories and composition, I look for common patterns over time, and identify the most active sources and their most requested targets. Eventually, I show that the activity of heavy-hitter sources is visible on all the daknets, no matter their geographical location and address space. In order to observe more relevant and less common events, anyway, the observation time and choosing the proper darknet size are two fundamental aspects.

Chapter 5 enriches the traffic characterization including different kinds of active probes. I compare three categories of active deployments, namely the L4-Responders, the L7-Responders and DPIPot, against the darknet. The L4-Responders is only able to establish a three way handshake with the remote source, without offering any application layer service but recording any received payload; the L7-Responders includes a set of well-known low-interaction honeypots, offering application layer services on their standard ports; DPIPot decouples the required service from the port, recognizing the required application by means of Deep Packet Inspection. I

show how actively engaging with the remote sources significantly increases the visibility of the darknet. I record a relevant increase in traffic volume (up to hundreds of times) also on neighbouring *dark* addresses. I quantify the *Side-Scan* phenomenon, in which a host responding on a restricted set of ports, related to a service, sees also a surge of traffic for other services and ports. Moreover, I observe how the ability of DPIPot to decouple services from ports sheds light on large-scale activities directed to non-standard ports.

After having characterized the macroscopic phenomena on the different deployments, I need to make a step forward towards the data analytics phase. Chapter 6 provides a methodology to automatically uncover communities in unsolicited traffic. A community is defined as a set of sources performing similar activities (e.g., requesting similar services or sequences of ports). Identifying common behaviors in traffic is a particularly relevant task, as it may help isolating botnets or other periodical scanning activities, which may otherwise remain unrelated. I first represent the activity of the remote sources toward the destination ports by means of a graph, on top of which I test several community detection algorithms. I show that off-the-shelf techniques as the Greedy Modularity Algorithm can successfully isolate communities performing vertical (i.e., specifically requesting only a few ports) or horizontal (i.e., widespread testing all the ports) scans, both in the darknet and in the honeypot case.

Together with community detection, another paramount task when dealing with unwanted traffic is the detection of anomalies. A surge in the traffic volume, for instance, may indicate the rise of a new botnet, of a misconfigured host performing repeated requests by mistake, or of a particularly aggressive new actor that may be worth isolating or blacklisting. Chapter 7 provides an overview of some of the most common anomaly detection algorithms, comparing them against a more recent one, the autoencoder. I evaluate these techniques, testing them first on two artificial labelled datasets, and eventually on the traffic collected on DPIPot. I define a set of metrics to be used in lack of a ground truth, and provide a set of practical considerations on the ease of implementation and on the visualization of the results.

Finally, Chapter 8 concludes the thesis and summarizes the next steps of my research.

Chapter 2

Related Work

Unwanted network traffic is commonly used as a fruitful source of information for network monitoring. In this Chapter I collect the most relevant papers and research projects addressing this topic, grouping them into categories according to the specific problem they target.

Figure 2.1 summarizes the taxonomy. I collect papers and projects dealing with fully passive probes (i.e., darknets) as well as various categories of active probes (i.e., honeypots). For both cases, some of the surveyed papers describe the way in which the system has been deployed, i.e., sparse sensors or contiguous IP blocks in the darknet case, or vertical, service-specific honeypots vs. horizontal ones. Other works characterize events on darknets or honeypots, describing for instance the rise of new threats and botnets, attack patterns and malicious login attempts, legitimate network scans, backscattering traffic generated by victims of attacks with spoofed IP addresses, or misconfigurations. Eventually, many papers present Machine Learning applications aimed at extracting knowledge from darknet traffic e.g., detecting communities, anomalies, finding recurrent patterns, etc.

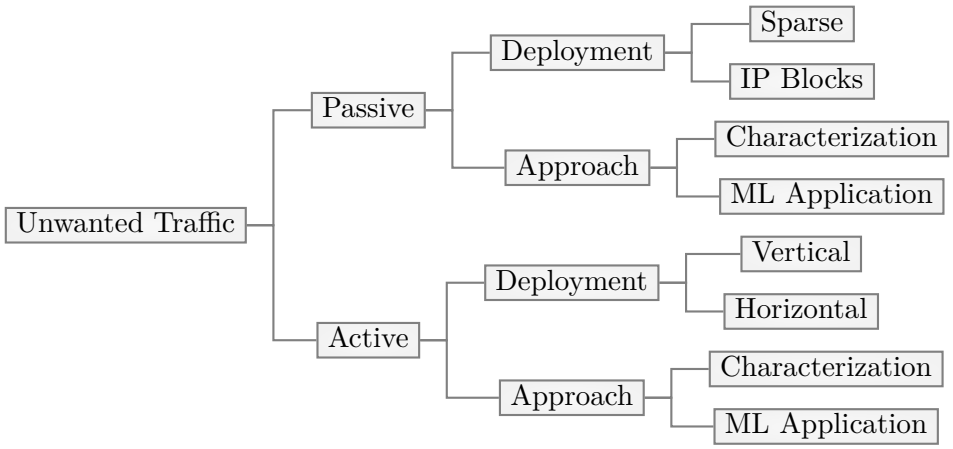


Figure 2.1: Unwanted traffic measurements taxonomy

Table 2.1: Summary of surveyed papers.

	Interactivity		**Focus**		**Objective**
	Passive	Active	Deployment[1]	Approach[2]	
[31]	✓		IP		/8 darknet deployment
[155]	✓		IP		/8 darknet deployment
[68]	✓		SP		Greynet run by company
[71]	✓		SP		Greynet run by university
[58]	✓		IP	CH, ML	Survey on darknet usage
[139]	✓		IP	CH	Characterize malware spread
[54]	✓		IP	CH	Characterize scans
[46]	✓		IP	CH	Characterize scans
[55]	✓		IP	CH	Characterize scans
[105]	✓		IP	CH	Characterize DDoS
[79]	✓		IP	CH	Characterize DDoS
[57]	✓		IP	CH	Characterize DDoS
[44]	✓		IP	CH	Characterize Internet censorship
[47]	✓		IP	CH	Characterize IPv4 usage
[112]	✓		IP	ML	Graph mining
[2]	✓		IP	ML	Markov chains
[117]	✓		IP	ML	Markov chains
[125]	✓		IP	ML	Markov chains
[107]	✓		IP	ML	Visualization techniques
[93]	✓		IP	ML	Community detection
[92]	✓		IP	ML	Community detection
[122]	✓		IP	CH	Unwanted traffic in production
[74]	✓	✓	IP, H	CH	Analysis of two-phase scanners
[143]		✓	H		Distributed honeypot system
[48]		✓	V	CH	Honeypot made of Facebook pages
[97]		✓	H		Honeypot mimicking mobile device
[152]		✓	H		Honeypot on mobile network
[70]		✓	H		SDN-based Honeynet deployment
[149]		✓	V	CH	Low- and medium-interaction honeypot fingerprinting
[108]		✓	V	CH	Low- and medium-interaction honeypot fingerprinting
[158]		✓	H		L-4 honeypot system
[18]		✓	H	CH	Characterization of events on honeypot system
[109]		✓	V,H		Survey on existing honeypot projects
[8]		✓	V	CH	Events on high-interaction honeypot
[142]		✓	V	CH	Ethereum honeypot
[141]		✓	H	CH	Comparison of honeypot activity across locations
[17]		✓	V,H	ML	Honeypot pattern identification with Markov Chains
[29]		✓	V		Honeypot mimicking cloud environment

[1]**Deployment:**
SP (Sparse), IP (IP Blocks), H (Horizontal), V (Vertical).
[2]**Approach:**
CH (Characterization), ML (Machine Learning).

Table 2.1 summarizes the content of each of the surveyed papers. Different papers describe the deployment of darknet infrastructure, from the large-scale projects run by the CAIDA/UCSD [31] and Merit [155] (each relying on a /8 IP range) to "sparse" darknets (also called greynets) run by companies [68] and academics [71]. The latter are characterized by a limited number of IP addresses that are distributed across different IP ranges. Several deployment strategies are thus available, and knowing the trade-offs is important for increasing the visibility of events

while reducing the allocation of addresses for darknets, particularly relevant given the shortage of IPv4 addresses.

More in detail, darknets have been used for a number of tasks [58], including (i) the investigation of malware spread [139] and Internet scans [54, 46, 55]; (ii) the estimation of DDoS frequency and volumes [105, 57, 79]; (iii) the analysis of Internet censorship [44]; (iv) the estimation of IPv4 address space utilization [47].

Given the enormous amount of traffic requests daily reaching the darknet sensors, finding a meaningful way of representing events is particularly difficult. Many previous works focused on detecting coordinated activities with the most diverse techniques. In [112], authors suggest to represent network traffic as a bipartite graph linking IP sources to /24 destination networks. Other works suggest to represent generic network data by means of first order [2, 117] or second order [125] Markov Chains. Authors of [123] propose a tool to ease the visualization of scanning activities, while [40] focus on topological analysis. Community detection is often used to evaluate social networks, but some works apply the algorithms to computer networks too. Authors of [107] separate legitimate and unsolicited email traffic. Authors of [93, 92] focus on Internet scans, characterized with an event-based graph defined as the sequences of ports contacted by scanners.

The analysis of traffic reaching fully passive sensors, however, carries some intrinsic limitations due to the impossibility to actively engage with remote sources. Moreover, a recent work [122] studying unsolicited traffic on Akamai's Content Delivery Network (CDN) suggested that the presence of production addresses close to the "dark" ones also influences the way passive sensors are contacted. Placing active honeypots close to darknet deployments may therefore have an impact on the behaviors we observe and deepen our knowledge of network events. A first interesting trade-off between completely passive and active deployments can be found in [74]: authors characterize the two-phase scanners phenomena, i.e., scanners firstly sending a first irregular TCP packet, then continuing with a regular TCP SYN after having received a response by possibly vulnerable targets. This analysis is lead by means of a *reactive network telescope*, defined as a /24 network capable of responding at transport level.

Application layer honeypots, on the other hand, have been used in security activities for years, with well-established projects such as the Honeynet Project [75] and TPot [143] providing multiple alternatives. Previous efforts using honeypots have covered many aspects, such as (i) introducing new honeypots targeting particular protocols, services or device types services [48, 97, 152], (ii) evaluating the effectiveness of different types of honeypots [70], and (iii) presenting techniques to uncover the presence of honeypots [149, 108]. A seminal work dating back to 2004 [158] introduced iSink, a monitoring system that can answer darknet traffic. The authors observed traffic peaks due to the presence of responders (e.g., NetBIOS and SMB) in the dark space. Later, another work [18] characterized honeypot traffic and discussed whether bots are sensitive to the "liveness" of hosts during scanning.

The authors found that in around 16.3% of the scan events, there was evidence of liveness-awareness. Readers are invited to check this survey [109] showing a broad overview on honeypot research.

Some authors present general characterization of honeypot traffic, focusing on origin of attacks, targeted services and frequency of attacks (e.g., [8, 142, 97]). A recent work [141] compared the deployment of honeypots in different geographic locations. Authors of [17] identify patterns used by attackers contacting honeypots. Another work [29] allocated unused addresses in a cloud to deploy honeypots. These works however evaluate honeypot deployments in isolation, without comparing measurements with what is observed in dark spaces.

Table 2.2: Summary of contributions for each chapter.

Chapter	**Contribution**
Chapter 4	Characterization of darknet traffic and similarity analysis for darknets deployed in different geographical locations
Chapter 5	Deployment of comprehensive framework for darknet, L4-Responders, L7-Responders and DPIPot
Chapter 6	Coordinated activity detection based on graph mining and community detection
Chapter 7	Benchmark of algorithms for anomaly detection in network traffic time-series

This thesis leverages the lessons learned from previous works to provide a comprehensive framework for the inspection and analysis of unwanted traffic. Table 2.2 provides a short summary of the original contributions brought by each chapter. Chapter 4 characterizes traffic hitting darknet targets around the world, with the aim of understanding how generalizable are the findings previously observed on darknets. Chapter 5 provides a thorough description of a newly deployed flexible infrastructure, including both passive and active probes replying to requests at different levels, enlarging the set of possibly observable phenomena om darknets and honeypots. Chapters 6 and 7 describe and extend several analysis techniques based on big-data and machine learning that contribute to enhance the knowledge on network phenomena, spot possible malicious users and link together apparently unrelated events. The framework as a whole allows a 360-degree view on network events, and aims at significantly easing and automating the work of network analysts and practitioners.

Chapter 3

Knowledge Extraction Pipeline

In this chapter I describe the pipeline required to collect and analyse darknet and honeypot data. At the end of the chapter, I briefly discuss the ethical implication of these operations and the countermeasures taken to preserve the privacy of possible victims of attacks.

3.1 Data collection and processing

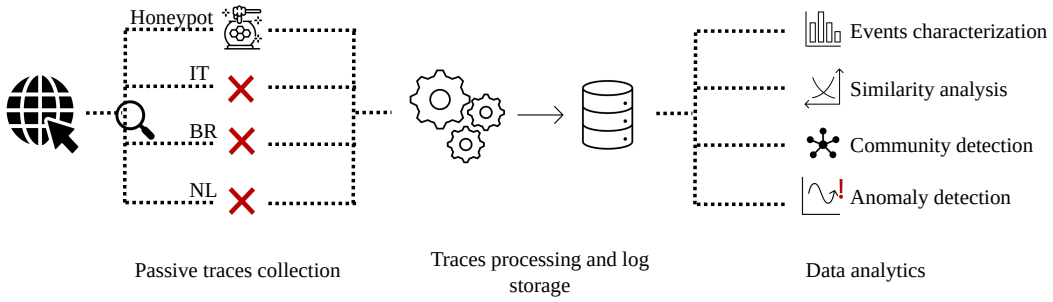


Figure 3.1: Data capture and processing pipeline

Figure 3.1 summarizes the data acquisition, processing and analysis procedure. I rely on three darknets located in two different continents: The first one is formed by a /15 network allocated by RIPE NCC in the Netherlands (hereafter called *NL*). The second one resides in Brazil, formed by a /19 network allocated by LACNIC (hereafter called *BR*). The third one is formed by three /24 networks, with non-continuous addresses, hosted at the Politecnico di Torino in Italy (hereafter called *IT*). This latter is particularly interesting, since the addresses were part of an active network until some years ago. IPv4 prefixes are kept private following requests of the research institutions running the networks. Together with the three darknets, an

honeypot infrastructure handling diverse protocols has been recently added to the Politecnico infrastructure (cfr. Chapter 5 and Appendix A for a detailed description of the honeypot deployment).

In each location a network probe captures the traffic arriving to the allocated address, recording the full packet. The probe obfuscates IPv4 prefixes of the darknets/honeypots (i.e., destination IP addresses) and sends the data to a Hadoop-based cluster for storage and processing. Traffic is stored on an hourly basis as a standard .pcap file. Table 3.1 reports the network size, total traffic volume and observation period for every infrastructure. As visible in the table, despite being active for a much shorter period of time, the amount of traffic hitting DPIPot, i.e., interacting with actively responding machines, is significantly larger than that hitting completely passive deployments. For passive deployments, the network size certainly plays a significant role on the received traffic volume.

Table 3.1: Dataset summary

	Network size	**Volume**	**From**	**To**
NL	/15	5.2 TB	Aug 14 2018	Jan 25 2019
BR	/19	1.3 TB	Sep 07 2018	Ongoing
IT	$3 \times /24$	70 GB	Dec 21 2018	Ongoing
DPIPot	$16 \times /29$	1.4 TB	Nov 12 2020	Ongoing

As a second step, I process the captured traces to generate human-readable logs. I use Tstat [144], a passive traces analyzer that processes every flow and enriches it with more than 200 useful measures (e.g., flow completion time, requested application protocol, Round Trip Time, etc.).[1] I further enrich the logs by adding the origin Autonomous System and Country. For this purpose I employ the pyasn library[2] and the MaxMind GeoLite2 Databases[3]. Note that in the darknet case, as the destination address is not replying to any request, I only record unidirectional, incomplete flows.

Eventually, I am ready to extract knowledge from the logs. Given the significant volume of traffic, the data analytics phase requires the usage of Big-Data techniques. I therefore run all the filtering and aggregation operations by means of pyspark[4] on an high-end Hadoop cluster. Each of the following chapters reports a more detailed description of the data analytics process. Chapter 4 provides a first characterization of the events on the darknet, together with a darknet similarity analysis; Chapter

[1] http://tstat.polito.it/measure.shtml

[2] https://github.com/hadiasghari/pyasn

[3] https://dev.maxmind.com/geoip/geolite2-free-geolocation-data?lang=en

[4] https://spark.apache.org/docs/latest/api/python/

5 analyzes the changes occurring on the infrastructure when new honeypots are active; Chapter 6 discusses the results of the community detection algorithms, and Chapter 7 benchmarks some anomaly detection techniques on the traffic data.

3.2 Ethics

When it comes to honeypot activity, several countermeasures need to be taken to restrict the impact of the measurements on third-party networks. First, and most important, the infrastructure never answer packets if this may worsen the position of attack victims. In particular, it never answers UDP traffic, as it could make the infrastructure itself part of DDoS attacks relying on spoofed addresses and amplification techniques. For the same reason, it silently drops all TCP packets with `SYN/ACK` flags and other malformed flows, as they may arrive from victims of DDoS attacks with spoofed addresses. Answering such packets may help the attackers to overload the victims' networks.

The analysed traffic may come from infected machines that are taking part in botnets. Therefore, I explicitly limit the capacity of the honeypot infrastructure to avoid creating too much traffic for the networks hosting such infected machines. The setup discussed in this thesis can comprehensively sustain at most some few Mbps of traffic upstream, which is far insufficient to overload remote networks.

Finally, IP addresses sending traffic to the infrastructure may uncover vulnerable computers exploited by attackers [134]. I take all measures to protect such IP addresses: In the following experiments, no IP address is disclosed. I also collaborate with the university security team and our upstream providers, actively notifying about novel attacks and senders.

Chapter 4

Are Darknets All the Same? On Darknet Visibility for Security Monitoring

The work I present in this chapter is mostly taken from my paper *Are darknets all the same? On darknet visibility for security monitoring*, presented at the IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN) 2019 [137].

4.1 Introduction

Darknets - also called network telescopes, Internet sinks and darkspaces - have been used for years as a source of information for cybersecurity [58]. A darknet is a set of IP addresses advertised by routing protocols, however without hosting any device. All traffic reaching the darknet remains unanswered and, by definition, is considered unsolicited. A monitoring probe listens to the darknet traffic, processing it in search for signals of new threats, misconfigurations and possibly sources/victims of attacks.

As highlighted in Chapter 2, darknets have been used for many diverse tasks, spanning from the investigation of internet scans and botnets, to the estimation of the address space utilization. Years of experience running darknets have shown that three main types of traffic reach such networks [22]: (1) networks scans, both malicious (e.g., by botnets) and legitimate (e.g., by crawlers and research projects); (2) backscattering, i.e., deflected traffic received because someone contacted a host spoofing the source IP address belonging to the darknet; (3) traffic due to misconfigured devices or mistyped IP addresses.

This chapter aims at understanding how the visibility of darknets varies according to the IP range, size and location of the darknet. I capture traffic simultaneously for 1 month in three darknets, deployed in the Netherlands (a /15 network), in Brazil (a /19 network) and in Italy (3 /24 networks). I contrast the traffic reaching each network, highlighting the mostly seen protocols. I confirm that the size of the darknet matters, and quantify how the visibility is affected by the number of IP addresses allocated for the monitoring. I show that the Autonomous Systems (AS) and countries originating the traffic present significant differences according to the IP range where the darknet is deployed as well as the considered time period. All in all, results show that darknet traffic must be used with care to support security tasks, since the picture obtained in one darknet may not reflect other darknets or the attacks seen on production networks.

Several studies already targeted the trade-offs in darknet deployment strategies. Since seminal works on Internet Background Radiation [106, 59] (i.e., traffic seen in darknets), authors question the impact of darknet size, an analysis that has been revisited some years later [155] and repeated for IPv6 [42]. Recently, authors of [22] compare traffic observed in CAIDA's and Merit's darknets. Other authors have focused on distributed darknets [71, 15, 38, 14]. All acknowledge that darknets deployed at different IP blocks and networks observe different events. These works are however aged, given the significant changes on the Internet in the last decade. I reappraisal this analysis with current traffic, shedding light on the coverage of interesting events according to parameters of the darknet deployment.

Table 4.1: Datasets and percentage packets per protocol.

	Size	Volume	TCP	UDP	ICMP	Other
BR	/19	2.5 GB/day	95.16%	4.39%	0.44%	0.01%
NL	/15	30 GB/day	93.69%	5.72%	0.59%	0.00%
IT	$3 \times /24$	420 MB/day	95.71%	3.89%	0.39%	0.01%

4.2 Methodology

I rely on data from three darknets described in Chapter 3. Given the different size of the darknets, most of the analyses of Section 4.3 are restricted to smaller subnets of *BR* and *NL* darknets (hereafter referred to as *NLs* and *BRs*, respectively) to allow a fair comparison with *IT*.

I perform analyses using data collected during 1 month, from the 1st of January to the 1st of February 2019.

Table 4.1 summarizes the dataset and provides a per-protocol breakdown of packets reaching the darknets. The majority of the traffic is represented by TCP

(> 93% of the packets), with UDP ranging in 3.89 – 5.72% and less than 1% for other protocols. No significant difference emerges between the darknets. The general picture is similar to the one reported in [22], even if higher percentages of TCP packets are present in these darknets with respect to what is reported in previous work.

When analyzing the composition of traffic reaching darknets, I will focus on some main *traffic categories*:

• **Scan:** TCP packets with only the `SYN` flag set. To filter occasional scan from actual hosts running extensive scans, I mark as scans only those cases where the sender targets at least $k = 10$ different destination addresses or ports in a one hour time bin;

• **Backscattering:** TCP packets with `SYN+ACK`, `RST`, `ECN`, `RST+ACK` or only `ACK` flags set. Since the darknet does not send any packets with `SYN` flag set, these packets are mostly from devices contacted with spoofed source IP addresses;

• **UDP:** UDP traffic, regardless payload or ports;

• **ICMP:** ICMP traffic;

• **Other:** All other cases or protocols. These include `SYN` scan messages sent by occasional scanners (that sent less than $k = 10$ messages in one hour).

4.3 Comparison of darknet traffic

In this section I provide a comparison across darknets, contrasting traffic composition, temporal patterns, sources and targeted ports.

4.3.1 Traffic types

Table 4.2 provides a breakdown of the traffic, showing the percentage of packets and unique source IP addresses distribution across categories. Considering the share of packets on different categories, the highest share of traffic is constituted by Scan, with small differences among the darknets. The *IT* darknet shows a lower share of backscattering because of middle-boxes sitting upstream the darknet, which drop incoming packets with inconsistent TCP flags/handshakes. UDP and ICMP shares are consistent among the three networks.

When comparing source addresses per category, interesting considerations hold. First, notice that Scan traffic is responsible of the majority of volume but it is generated by a small fraction (3.2 to 12.5%) of the senders' IP addresses. Recall that these sources are involved in non-occasional *SYN scans*, given the filters described in Section 4.2. Second, there is a larger number of distinct IP addresses sending UDP packets to *NL* than to *BR* and *IT*. Here manual inspection confirms a fact about darknet traffic [22]: There exist few sources that send a lot of UDP packets to targeted IP addresses. If the darknet does not include any of such targeted

Table 4.2: Summary of the traffic per category.

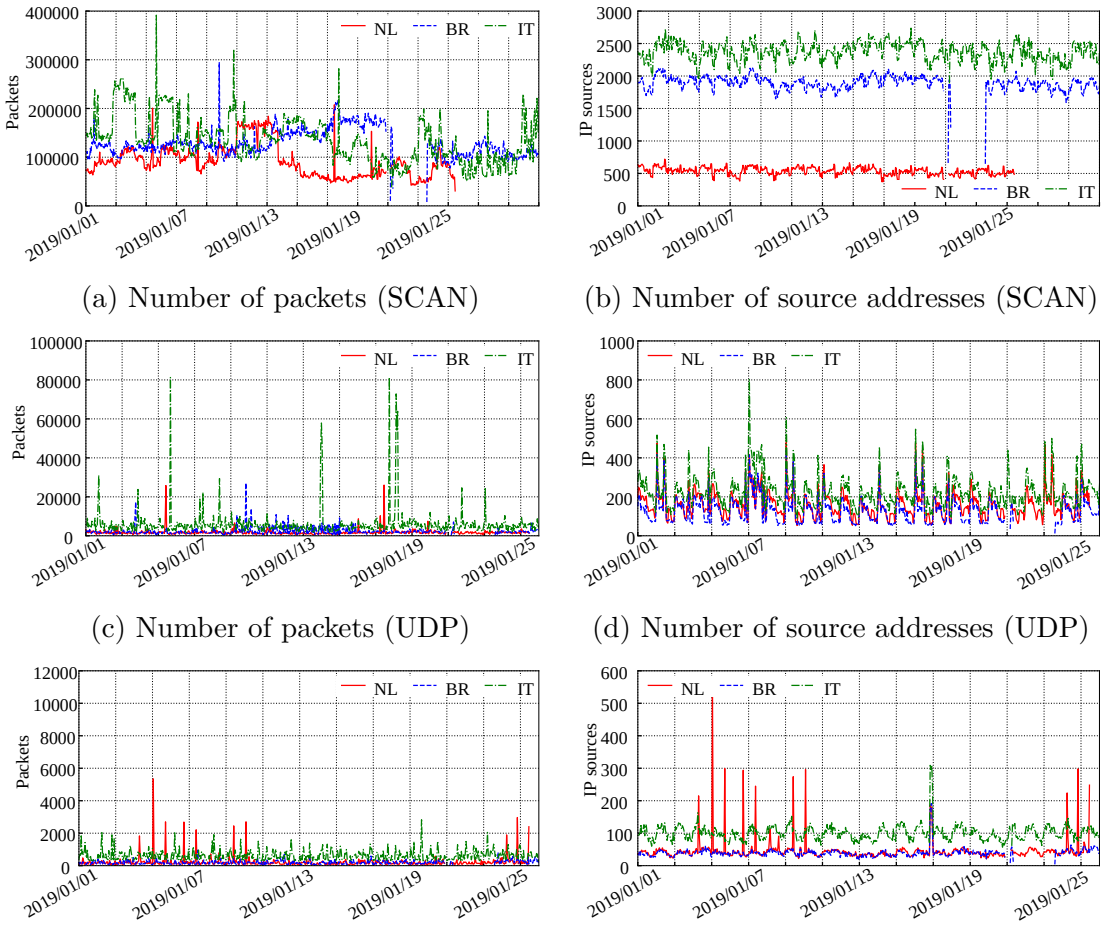
Type	*NL*/15		*BR*/19		*IT* 3 × /24	
	Pkts	IP addr.	Pkts	IP addr.	Pkts	IP addr.
Scan	85.1%	12.5%	84.8%	4.6%	86.9%	3.2%
Back.	3.7%	0.8%	2.3%	0.6%	0.2%	0.2%
UDP	5.7%	10.8%	4.3%	2.3%	3.8%	1.8%
ICMP	0.5%	1.6%	0.5%	0.8%	0.3%	0.6%
Other	4.8%	74.1%	7.8%	91.4%	8.6%	93.9%

destinations, it would see less UDP events. Finally, note the large percentages of sources whose traffic lies in the "Other" category. Recall these are occasional scans, where the sender sends only few packets to the darknet. This traffic may be due to misconfigurations, low-rate attacks, or stale information in e.g., P2P protocols.

4.3.2 Temporal patterns

I next check whether the traffic reaching different darknets follows similar temporal patterns. Since the darknets have different sizes, both the /15 *NL* and the /19 *BR* darknets have been split into smaller subnets having the same dimension as the Italian one - i.e., 3 × /24. In the remainder of this Section, I restrict *all* analyses to 3 Dutch and 3 Brazilian samples, thus allowing a fair comparisons with the Italian one. Figure 4.1 reports time series of packets (left) and IP sources (right) per hour for the most relevant traffic categories. The remaining categories are omitted given their lesser contribution to the total amount of traffic, and their noisy temporal pattern. Notice that the gap in the figures is due to a temporary failure in the Brazilian infrastructure.

Scan traffic (Figure 4.1a) presents no clear temporal pattern and no periodicity. Equally, there is no apparent similarity between *BRs*, *NLs* and *IT*, and traffic peaks do not appear to be simultaneous. Similar considerations hold for less relevant traffic types, i.e., UDP (Figure 4.1c) and ICMP (Figure 4.1e). For these categories the total number of hourly packets is 1 to 3 orders of magnitude lower than the SCAN case, with some occasional peaks, appearing with no periodicity. Figure 4.1b shows, instead, a more regular pattern in the number of distinct IP sources per hour. Notice that the number of addresses hitting *IT* is generally higher than the ones hitting *BRs* and *NLs*. I conjecture that this possibly happens because such addresses have been previously allocated to a production network, and may hence be more known. Similarly, the lowest number of sources is observed on *NLs*, whose addresses have always been allocated as a darknet space. This suggests that the *NLs* darknet may be known to attackers that avoid targeting it. Figure 4.1d, shows, instead an higher level of similarity in the patterns of sources hitting the different darknets, with the *IT* darknet still hit by an higher number of remote hosts. This



(a) Number of packets (SCAN)

(b) Number of source addresses (SCAN)

(c) Number of packets (UDP)

(d) Number of source addresses (UDP)

(e) Number of packets (ICMP)

(f) Number of source addresses (ICMP)

Figure 4.1: Time series (1h bins) for packets and sources.

result suggests that the hosts searching for UDP services and vulnerabilities are fewer, possibly the same, and they target a wider set of destinations, independently from their locations. ICMP (4.1d) is requested by some tens of sources in the *NL* and *BR* darknets. The patterns are similar across both darknets, with some peaks on *NL*. Also in this case, the *IT* darknet is targeted by more sources. Notice the simultaneous peak across the three darknet around the 17$^{\text{th}}$ January: this may suggest a common origin of the ICMP requests.

To confirm these hypothesis, I further analyze the time series by calculating the Pearson correlation coefficient between pairs of darknets. Considering the number of packets, for all traffic categories the pairwise correlation is zero or slightly negative – i.e., time series are uncorrelated. Different considerations hold for the number of sources: The average correlation among all pairs is 0.47 for network

scans, 0.31 for backscattering, with *IT* and *BRs* reaching 0.79, 0.87 for UDP and 0.42 for ICMP (again with a peak of 0.79 between *IT* and *BRs*).

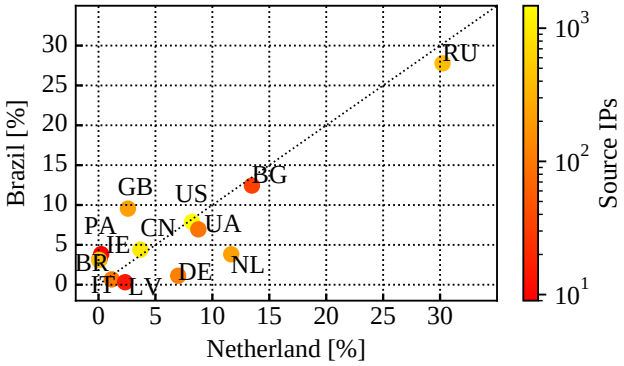


Figure 4.2: Top source countries for Scan traffic.

Table 4.3: Top-10 AS per SCAN traffic – Jan 2019.

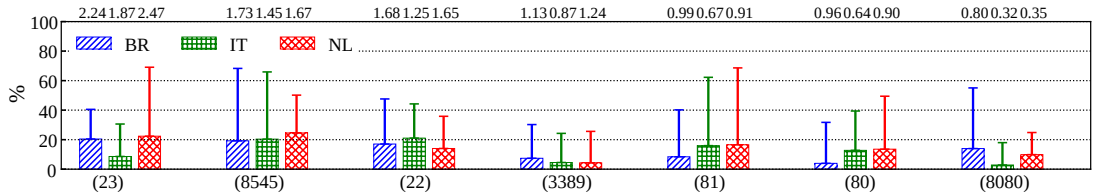
BRs			*NLs*			*IT*		
ASN	pkts	IPs	ASN	pkts	IPs	ASN	pkts	IPs
49453	14.8	8	49505	10.57	15	**43350**	22.18	12
57043	10.72	15	202325	9.94	11	204428	7.17	24
202325	6.5	12	204428	7.52	20	58271	7.05	22
58271	5.18	19	58271	6.9	19	**51852**	6.69	5
204428	3.74	18	**201912**	5.8	8	57043	6.28	16
14061	2.75	542	47350	5.07	5	14061	2.80	658
57271	2.51	11	57271	3.38	11	202325	2.75	11
47350	1.86	8	14061	3.03	103	**202425**	2.03	45
50297	1.66	4	**48817**	2.17	8	**206485**	1.74	3
51787	1.64	4	**41390**	1.96	1	49505	1.63	27

4.3.3 Origin of Scan traffic

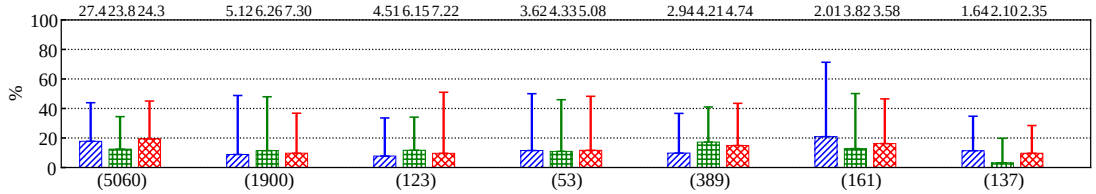
I now focus on Scan packets to check whether sources of traffic are similar across darknets. Beside considering source IP addresses, I also map them to the corresponding AS and country with the Maxmind Geo Location database.[1]

Considering IP addresses, I record 27,105 sources for *BRs*, 29,837 for *IT* and 4,269 for *NLs*. As previous works observed, the distribution of packets per IP

[1] https://www.maxmind.com/en/home



(a) Scan



(b) UDP

Figure 4.3: Packets due to top-1 (boxes) and top-10 (whiskers) source IP addresses for the 7 most contacted TCP and UDP ports. Numbers in the top x-axis represent the share of the port in the overall number of packets for the given network.

address is heavy tailed: in our case 95% of the packets are generated by the (i) 22% most active addresses in *BRs*, (ii) by the 18% most active addresses in *NLs* and (iii) by the 23.3% addresses in *IT*. Notice the different order of magnitude in the number of sources observed in *NLs* with respect to the other two darknets. This result leads to the same conjecture reported in Figure 4.1b: being *NL* addresses allocated in the darknet space from a long time, they may be known as unused and hence less targeted.

Considering source ASes, I find 1,393 (*BRs*), 142 (*NLs*) and 1,524 (*IT*) sources, of which 134 are common to all three darknets (i.e., more than the 94% of the addresses targeting *NLs* are seen also in *BRs* and *IT*), while 1,015 are common between *BRs* and *IT* (i.e., the 72.8% of the ASes seen in *BRs* are also present in *IT*). The top-10 most active ASes are shown in Table 4.3, which reports in bold those that are not common across the three darknets. Table 4.3 also shows that the most common ASes generally produce a large percentage of traffic using only a small set of addresses (with AS 14,061 being the exception). Moreover, rarely a single AS targets all darknet in the same way – e.g., AS 49,450 is the most active against *NLs*, but it is ranked as last in *IT*, even if the latter is targeted using a wider number of sources.

I finally focus on source countries. In total 133 countries are seen on *IT*, 125 on *BRs*, and only 38 on *NLs* (the latter are all visible also in *IT* and *BRs*). Figure 4.2 compares the top-10 most seen countries per *BRs* and *NLs*. The scatter plot compares the share of packets from each country, while colors mark the total number of

IP addresses observed for the country. The ranks mostly overlap, with 13 countries building the combined lists. Russia is the most popular source for both *BRs* and *NLs*, together with Bulgaria, Great Britain, USA, Ukraine and China.

In general, such results confirm a conjecture raised in [22]: The Scan traffic reaching different darknets, while similar, is non-uniform.

4.3.4 Per-port breakdown

I now examine the destination ports of packets reaching the darknets. We restrict our analysis to Scan and UDP traffic since, for backscattering, destination port does not contain useful information.[2] Again, I consider only the three /24 *NLs* and *BRs* subnets for a fair comparison with the *IT* darknet.

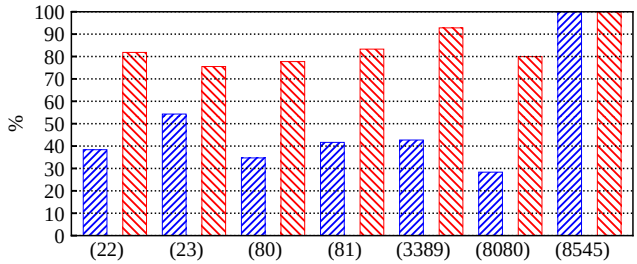
In Figure 4.3 I quantify to what extent traffic originates from a small or large set of addresses for the 7 most contacted ports. Boxes represent the share of packets sent by the single most active IP source, while the whiskers represent the share for the top-10 addresses. Numbers in the top of the figure report the overall percentage of traffic to the port in the given network.

Considering Scan (Figure 4.3a), the most popular ports are associated with services known to be targets of attacks, e.g., telnet (port 23) and ssh (port 22). A significant number of such packets have been linked to attacks targeting IoT devices [101]. The top-10 sources are generally responsible for less than the 40% of the traffic, except in some particular cases (for instance port 81 and 8545, which is targeted by less distributed sources). The single most active source is in most cases generating about the 20% of the traffic alone. Focusing on the upper x-axis, I notice that the volume hitting the ports is similar across darknets.

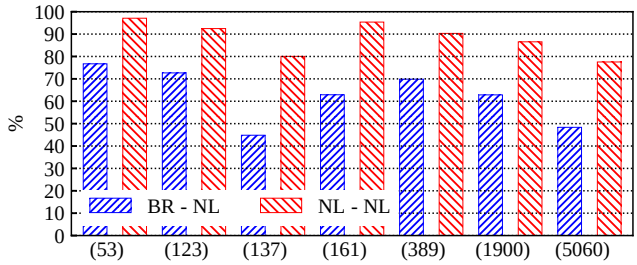
Focusing on UDP (Figure 4.3b), I see that for almost all ports, about 40% of the packets are related to top-10 IP addresses. Pictures emerging in the three darknets are very similar, with protocols such as SIP (port 5060), NTP (port 123) and UPnP (port 1900) leading the ranks. This is not surprising, as such protocols are well-known targets large-scale attacks and abuses. Again, focusing on the upper x-axis, I notice that a similar volume of traffic hits the considered ports for the three darknets.

I finally quantify to what extent the traffic sources are shared among darknets. Again, I map each IP address to the corresponding AS, and, separately per port, compute the Jaccard similarity index between the obtained sets of ASes. We consider *NLs* and *BRs* darknets, as they resides in different continents. Figure 4.4 shows the results. The blue bars represent the average similarity when comparing *BRs* to *NLs* subnets. The red bars serve as baseline, showing the average similarity when comparing the three *NLs* subnets against each other. Considering Scan,

[2] Backscattering traffic typically comes from victims contacted with spoofed source IP address.



(a) Scan



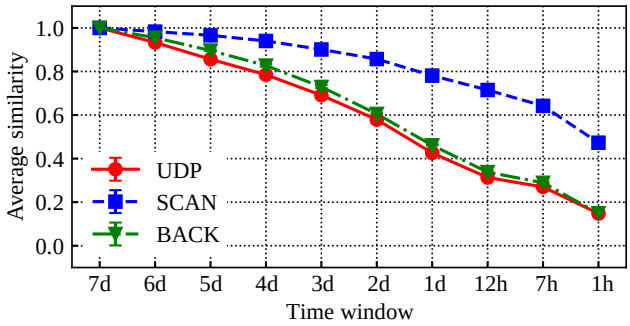
(b) UDP

Figure 4.4: Average Jaccard similarity (calculated over sets of ASes) between *BR* and *NL* samples (blue) and among *NL* samples (red) for the top contacted ports.

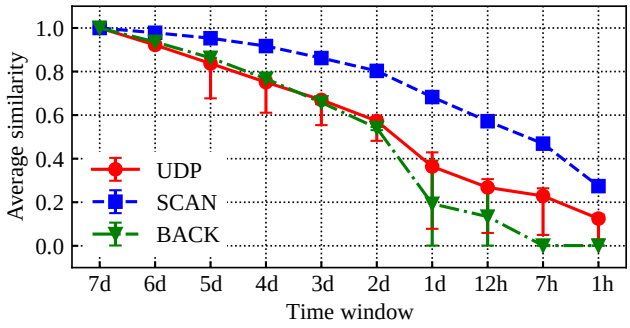
Figure 4.4a shows a Jaccard always around the 80% for *NLs* subnets, as expected. The similarity is generally below 50% when comparing *BRs* to *NLs*. Very interesting is the case of port 8548 (Json-RPC), which shows that scanning attempts on *BRs* and *NLs* descend from exactly the same set of ASes. The same considerations hold for the UDP scenario when comparing the *NLs* subnets among themselves. In Figure 4.4b, I see that the Jaccard index is generally above 80% when comparing the *NLs* subnets, while, when comparing *BRs* and *NLs*, results vary from port to port (e.g., below 50% for port 137 and port 5060, above 70% for port 53 and 123). The high values of Jaccard Index for ports 53, 123 and 389 may further explain the similarities in patterns I highlighted in Figure 4.1d: some of the most requested UDP ports are targeted by the same sources, regardless of the geographical location of the destination.

4.4 Effects of darknet size

In this section, I verify how the observation period and the darknet size affect the list of observed sources. As in the previous section, I rely on the Jaccard similarity to compare the setups, focusing on source ASes.



(a) *NL* (/19 samples)



(b) *BR* (/22 samples)

Figure 4.5: Average Jaccard similarity when fixing the darknet size and varying observation time.

4.4.1 Observation period

I first analyze the impact of the observation period. For a given darknet setup, I extract the set of ASes observed in a 1-week long period. Then, I again extract the sets of ASes after reducing the observation period to given shorter intervals. The Jaccard similarity is calculated by comparing the sets obtained with short intervals against the one obtained with the 1-week long interval. To increase reliability on results, these steps have been performed multiple times, by sampling 8 /19 subnets from the original /15 *NL* darknet, and 8 /22 subnets from the /19 *BR* darknet. In Figure 4.5, each data point reports the average Jaccard similarity for the multiple subnets.

Figure 4.5a reports results for *NL*. Separate lines depict results for network scans, backscattering and UDP traffic. The visibility of sources in a darknet is reduced considerably when the observation period is reduced. However, the impact is different for the different traffic types. Considering network scans, the Jaccard similarity is still at around 0.8 when the observation period is reduced to 1 day (i.e., 80% of

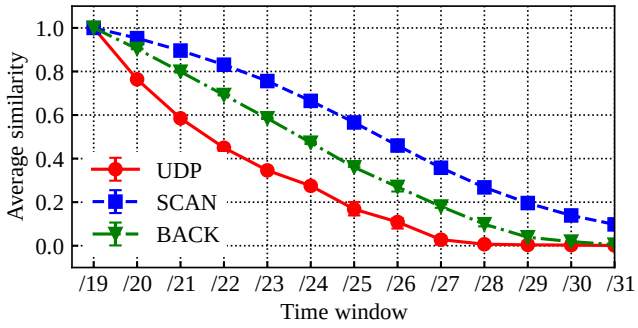
the sources seen in one week are visible in one day despite the reduced interval), and around half of the ASes are found if one observes only few hours of traffic. Instead, for UDP and backscattering traffic, the reduction on visibility is much sharper. Already after shrinking the observation period to 2 days, almost half of the ASes are lost. This is also a consequence of the overall volume per traffic type (see Table 4.2): whereas network scans are widespread, the other categories are rarer, which thus need more time to be observed.
Similar considerations hold for *BR* (Figure 4.5b). Given the smaller dimension of the darknet, the decrease is faster. For network scans, the picture is similar to the *NL* case, with just 30% of the sources found with a 1 hour observation interval. For backscattering, I notice a sudden drop when the observation period is shorter than 2 days. This can be explained by the intrinsic variability of backscattering traffic. Remind that backscattering sources are likely to be victims of attacks with spoofed addresses, typically carried out in a short amount of time. Given the lower volume, UDP decreases faster, too.

4.4.2 Darknet size

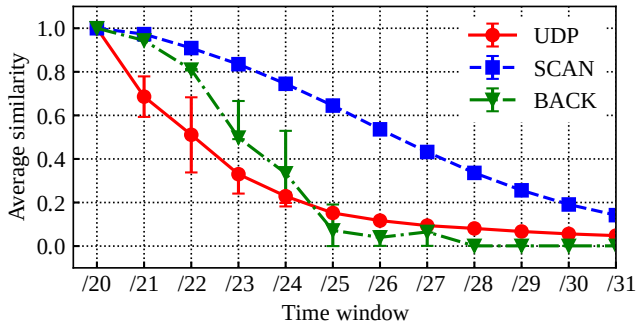
Finally, I analyze the impact of the darknet size, quantifying to what extent a small darknet observes events also found in larger ones. Remind that a small darknet would require low numbers of IPv4 addresses, thus freeing addresses for production traffic. Figure 4.6 reports the average similarity obtained when reducing the darknet size. Again, for improving robustness of results, we start by taking samples from the original darknets, i.e., 8 /19 subnets for *NL* and 2 /20 for *BR*. Then, for each experiment, we split each of these subnets into smaller subnets.

The figure reports the average similarity when comparing the small subnets to their respective original /19 (*NL*) or /20 (*BR*) subnets. The observation time window is fixed to one week in all cases.

Considering *NL* (Figure 4.6a), I notice a regular decrease as darknet size shrinks. For network scans, the plot suggests that a /25 network still observes around 60% of the source ASes. To obtain the same result for backscattering and UDP, larger /23 and /21 are needed. In other words, the majority of sources is still visible with a 64-fold reduction in darknet size for scanning, while for backscattering and UDP already a 16-fold and 4-fold size reduction hides 40% of the sources, respectively.
Similar considerations hold for *BR* (Figure 4.6b). For scanning, with a 64-fold size reduction (/26 subnet) more than half source are still present. UDP decreases slightly faster than the *NL* case, with a 8-fold size reduction (/23 subnet) already hiding almost 70% of sources. For backscattering, again I notice a much faster decrease with respect to *NL*, similarly to what emerged from Figure 4.5b. The /25 subnets lose almost all visibility, confirming that the variability and unpredictability of backscattering traffic requires large darknets to be observed.



(a) Netherland



(b) Brazil

Figure 4.6: Average Jaccard similarity when fixing observation time and varying the darknet size.

4.5 Conclusion

In this chapter I compared three darknets deployed at different IP ranges and continents. I confirmed well-known facts about darknet visibility, such as the prevalence of traffic to the ports usually targeted by scans and attacks. I observed that the largest amount of traffic is produced by few source IP addresses performing massive Scan attempts, while most of them send only few packets. The behaviour of sources is particularly similar across darknets for UDP traffic, while lower similarities are registered for other traffic categories. The top contacted ports are similar, and mostly composed of well-known targets. For some of them, only few source ASes are behind the traffic. However, the ASes targeting most ports vary considerably along the observed time period.

Moreover, I pointed out that network scans are constant and more prominent, thus easier to monitor. A one order of magnitude reduction in observation time and IP range size removes little of the darknet visibility. For backscattering and UDP, large observation times and IP ranges are needed for a good coverage.

All in all, these results provided new evidences that sources of traffic significantly

varies according to the IP range, and the size of the darknet impacts its visibility.

Chapter 5

Enlightening the Darknets: Augmenting Darknet Visibility with Active Probes

The work I present in this chapter is mostly taken from my paper *Enlightening the Darknets: Augmenting Darknet Visibility with Active Probes*, currently under evaluation at IEEE Transactions on Network and Service Management.

5.1 Introduction

Darknets or network telescopes are IP addresses advertised by routing protocols without hosting any services. As already stated in the previous chapters, they have been used for years as *passive sensors* in a variety of network monitoring activities. Darknets have supported multiple research activities too [58], and multiple organizations worldwide deploy, analyze, and eventually archive years of data from large darknets for research purposes [22, 155, 46]. Traffic reaching a darknet is necessarily unsolicited. As such, it is helpful to highlight network scans (from both malicious and legitimate scanners) backscattering (i.e., traffic received from victims of attacks carried out with spoofed IP addresses) and traffic due to bugs and misconfigurations [22].

The visibility of darknets is intrinsically limited, as traffic reaching darknets is never answered. Researchers, operators and practitioners have often relied upon external *honeypots* to obtain further information about events seen in darknets [12, 101, 79]. Unlike darknets, honeypots are *active sensors* that obtain information about attacks by answering unsolicited traffic. The goal is to engage with the possible attackers using both simulators that reproduce basic functionalities of real systems (low-interaction honeypots) or actual live system deployed in controlled environments (high-interaction honeypots). Typically, such honeypots are deployed

as vertical systems, targeting a particular scenario [109] and exposing only a single - or just a few - services. For instance, database or terminal server honeypots [52, 41] supporting the respective protocols are deployed on the standard service ports. Only few honeypots try to dynamically determine protocols present in the incoming traffic on-the-fly [65, 76].

Darknets and honeypots are complementary: The first offers a broad but shallow view on ongoing scanning activity touching a wide spectrum of services; The second offers deeper insights about specific cases. Combining "the best of both worlds", achieving at the same time wide coverage and deep insights, could enrich the type of information currently extracted from darknets. Indeed, it is known that darknet traffic changes substantially, not only across the IP address space, but also because of production services hosted "nearby" the darknet [122, 22, 137]. As such, the deployment of active services *inside* the darknet (e.g., using honeypots) could represent a novel perspective to enrich characterization of unsolicited traffic. However, such a deployment also raises many questions: (i) How much *extra information* one would get when responding some unsolicited darknet traffic? (ii) How does the presence of active services change *ports and senders*[1] seen in different deployments? Does the presence of these services affect *neighbouring darknet addresses*? (iii) Does the presence of a specific service attract traffic to *other services*? (iv) What if one answers any service on *non-standard ports*?

Having these questions in mind, in this chapter I quantitatively compare different levels of interactive responders deployed in a darknet address space. In particular, I consider the following four levels:

- Darknet, silent listeners that capture received traffic.
- L4-Responders, that only complete the TCP handshake, saving any possible application-layer requests sent by clients.
- L7-Responders, low-interaction honeypots that mimic specific application protocols that are expected on the usual well-known ports.
- DPIPot, a novel responder that searches for the application that is most suited to handle the incoming traffic, regardless of the probed port.

I capture data reaching these deployments for one month: in these measurements, I observe traffic volumes on the order of *hundreds* (on the darknet) to *millions* (on DPIPot) of flows per hour, collecting more than 1 *billion* flows overall. Digging into this dataset, I show that active responders can increase the value of darknet data and provide a detailed quantitative view on the behavior of bots and

[1] I use the generic term "sender" to indicate attackers, scanners, or even victims of attacks that send traffic to the infrastructure.

scanners when facing responders with different interaction levels. In some cases, I revisit and update well-known facts about darknet/honeypot deployments. Most importantly, I gather novel insights that highlight the benefits and drawbacks of the alternative response strategies. Summarizing my key findings:

- I measure and quantify *Side-Scan* phenomena, in which a host responding on a particular service sees also a surge of traffic for other services and ports. In contrast to [122], who observe similar patterns in CDN nodes, I quantify how the type of service one deploys in the darknet decisively influences *Side-Scan*.
- Activating responders leads to a surge of traffic on darknet neighboring IP addresses too. The joint use of active probes and passive darknets therefore leads to an overall picture that is different from what is obtained by a pure darknet deployment, even in the addresses remaining dark.
- L4-Responders augment the visibility of darknets, triggering senders to send additional traffic that uncovers malicious activities in some cases. Interestingly, the lack of application-layer response, while limiting the interaction, also uncovers events that may get unnoticed on classic honeypots, such as "port-knocking" attempts.
- L7-Responders engage senders and attract hundreds of times more traffic to the darknet. Deploying different L7-Responders to cover specific service categories helps to increase the *Side-Scan* phenomena, thus augmenting the value of darknet data.
- Thanks to the ability of DPIPot to decouple services from ports, I shed light on large-scale activities directed to non-standard ports, offering a rich picture that is unseen in other deployments. However, it may trap senders in some in particular activities, saturating them and thus limiting visibility. This trade-off suggests a complementarity between completely passive and totally interactive deployment.

I detail the infrastructure deployment and main features (Sec. 5.2), and the design of experiments and deployment (Sec. 5.3). I then provide a generic overview of the macroscopic changes I observe (Sec. 5.4), then I investigate the changes in the contacted ports (Sec. 5.5.1) and senders contacting the different deployments (Sec. 5.5.2). I then drill down on the benefits of DPIPot (Sec. 5.6) before final discussions and conclusions (Sec. 5.7).

5.2 Infrastructure

Fig. 5.1 schematizes the measurement infrastructure. Unsolicited traffic that reaches the dedicated address space is routed to one of the four *deployments* that

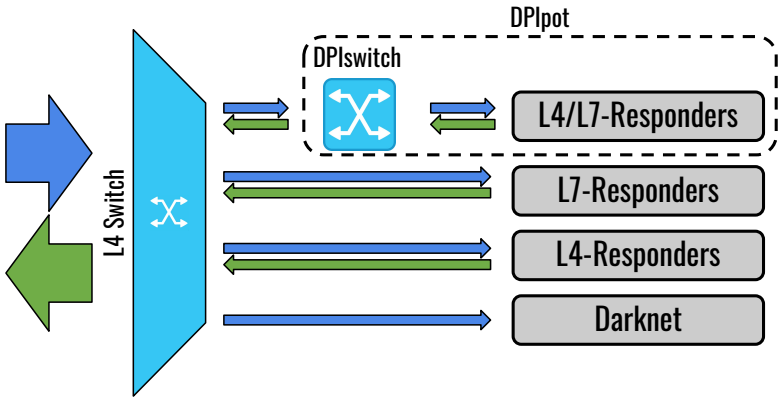


Figure 5.1: Infrastructure architecture overview.

correspond to different levels of interactivity:

1. Darknet: IP addresses that just receive traffic without replying to any packet;
2. L4-Responders: responders that complete the TCP three-way handshake, capture eventual application requests from clients, but never reply to any message;
3. L7-Responders: honeypots that mimic popular application-layer services. I deploy state-of-the-art honeypots to simulate well-known services; L7-Responders act as vertical responders that interact only on a limited set of ports and services;
4. DPIPot, a novel responder that performs L7 switching of requests using deep packet inspection. It decides on-the-fly which application protocol to use, answering incoming TCP connections on all TCP ports. Differently from L7-Responders, DPIPot decouples standard TCP ports from application protocols.[2]

Note that none of our deployments answers to UDP or malformed TCP packets (e.g., packets showing syntactically incorrect flags) for preventing abuses. Both the L4-Responders and DPIPot are implemented in Python using the Twisted framework [146]. The architecture (see Fig. 5.1) is intrinsically distributed, and Twisted is capable of handling a large number of connections at the same time. However, as I will show later, the deployment of responders increases the traffic reaching the darknet by orders of magnitude.

[2]To avoid resource starvation, both our L4-Responders and DPIPot implement active and inactive timeouts, dropping active (idle) connections after 60 s (10 s).

For the L7-Responders deployments, I rely on the honeypots organized and distributed by the TPot project [143] that act as backend. I activate honeypots to handle a range of popular application protocols. TPot offers a collection of third-party *low-interaction* honeypots, i.e., programs crafted to simulate a vulnerable service communicating over an L7 protocol. Most of our L7-Responders offer login interfaces only [73], registering the brute-force attempts against services (e.g., RDP, POP3 and IMAP). Some L7-Responders rely on more sophisticated honeypots, e.g., simulating a vulnerable server accessible via SSH/Telnet [41], or serving pages that mimic actual services accessible over the web [133]. The L7-Responders offer *vertical services* only: They are deployed behind the standard TCP ports of the given service, e.g., the HTTP honeypot is deployed on port TCP/80 whereas the Remote Desktop Protocol (RDP) honeypot responds to port TCP/3389.

5.2.1 DPIswitch implementation

To study the impact of answering to traffic arriving on other ports, I implement and deploy DPIPot. DPIPot listens to *all* TCP ports. As shown in Fig. 5.1, a paramount component of the DPIPot deployment is the *DPIswitch*. On receiving a new TCP connection request, DPIPot completes the three-way-handshake and waits for the first message from the client. Before forwarding the new packet to the most appropriate backend, the DPIswitch analyses the payload looking for the application-layer protocol. As the name suggests, the DPIswitch integrates a Deep Packet Inspection tool to perform payload analysis.

The choice of the most suited DPI tool to be implemented in the infrastructure follows a careful evaluation of four well-known DPI solutions[3], namely nDPI [49], Libprotoident [10], Tstat [144] and Zeek[4] (formerly *Bro* [116]). I restrict the choice to these four tools, as they are open-source, flexible and well documented. The evaluation aims at determining which tool yields the best classification performance, and whether it can be used in online scenarios. I use real traffic traces covering various traffic scenarios, including operational networks, IoT scenarios, media and games, and malware traffic. As no ground truth is available, I study the consistency of classification across the solutions, investigating root-causes of conflicts. Important for on-line security applications, I check whether DPI solutions provide reliable classification with a limited number of packets per flow. All in all, I confirm that DPI solutions still perform satisfactorily for well-known protocols. They however struggle with some P2P traffic and security scenarios (e.g., with malware traffic). All tested solutions reach a final classification after observing few packets with payload, showing adequacy for on-line applications.

[3] More details about this evaluation are reported in Appendix A.

[4] https://zeek.org

The final choice fell on nDPI [49], as it proved to be the most accurate in most of the analysed scenarios. This choice lets us obtain a flexible system, supporting hundreds of protocols, which is far more than what is supported in previous projects [76].

If a known protocol is found and one of the L7-Responders can handle it, DPIPot steers traffic to such backend; otherwise it acts like L4-Responders. Note that DPIPot can only identify and steer traffic for cases that are *client initiated*, i.e., where the client sends the first application-layer message. Otherwise, it behaves like L4-Responders– e.g., in telnet or SMTP, where the client waits for the server banner before attempting to login.

Table 5.1: Service categories with their typical ports and applications. Note that services at application level are only available for L7-Responders and DPIPot.

Category	**Category: Port**	**Category: Application**
All	0:65535	*All below*
Database	3306, 33060,* 1433, 4022,*, 1434,* 5432,* 27017	mysql, mssql, postgres, mongodb
Fileserver	135:139, 445	netbios, CIFS
Mail	25, 110, 143, 465, 993, 995	pop(s), imap(s), smtp(s)
Proxy	8080, 8000,* 3128	generic, squid
Remote Desktop	3389, 5900, 5901, 5800,* 5801,* 5938,* 6568*	ms rd, vnc, teamviewer, anydesk
Terminal	22, 2222,* 23, 2323*	ssh, telnet
Web	80, 443	http(s)
None	0:65535	-

(∗) Ports that are forwarded to the L7-Responders, even if the backend (i.e., TPot) does not host any honeypot. The L7-Responders reset the connection in these cases, as opposed to the darknet (which never answers traffic) and the L4-Responders (which always try to open a connection request).

5.3 Methodology and datasets

5.3.1 Deployments and categories

Inside the regular /16 campus network that hosts servers and clients, I isolate one /23 network to perform experiments with our multiple *deployments* – i.e., darknet, L4-Responders, L7-Responders or DPIPot– in equal conditions. I split the /23 network into /29 networks, each with 8 IP addresses. I dedicate sixteen /29 networks (all belonging to the same /24 network and hosting 8 identical responders each) to L4-Responders and L7-Responders, answering to a specific service *category*. I configure 8 categories for L4-Responders and 8 for L7-Responders (see Table 5.1). The category defines which services the responder supports. I configure the responders to receive and handle only traffic that arrives to ports typically hosting services belonging to such category, silently dropping packets arriving on other

ports (e.g., as done by some firewalls). I create categories for database, file, mail, proxy, remote desktop, terminal, and web services. I report all ports opened for each category on Table 5.1, together with some typical applications relying on such ports. I also create an *extra* category denoted as *All*, for which the deployments accept all traffic going to any port. In the case of the *all* category in L4-Responders, the system performs TCP handshake for flows arriving in any TCP port. For the L7-Responders category denoted as *All*, the system forwards all traffic to the TPot backend, regardless on whether there is a honeypot active on that port or not: if no honeypot is present, the backend explicitly resets the connection.

I devote a /29 network to host DPIPot, which answers to all ports by definition. It performs DPI on the arriving packets to identify the most appropriate responder based on the payload, and eventually forwards traffic to a honeypot offered by TPot. The remaining IP addresses in the /23 network operate as a classic darknet. In particular, I configure an entire /24 network as a darknet, to serve as a baseline for my experiments. Unless explicitly mentioned, all results using a darknet as baseline refer to such addresses. The second /24 network partly hosts responders and partly acts as a darknet – these latter addresses are used in Sec. 5.5 to understand the impact of active services on neighbour darknet addresses.

5.3.2 Data capture and processing

The infrastructure captures all packets hitting the /23 darknet. I use Tstat and store all traces on a high-end server, generating separate logs for each deployment.

I here characterize the traffic focusing on TCP flows, defined by the usual 5-tuple (client/server IP addresses, client/server ports and transport-layer protocol). A new flow starts when a `SYN` segment is received, and it terminates after the connection is closed (in case of the active responders) or an idle time. I annotate each flow with useful metadata and statistics, including the application protocol identified by `nDPI`, if any L7 payload is present. Table 5.2 shows an overall traffic breakdown for each of our deployments and categories.

According to the capabilities of each responder and the behavior of remote machines, I identify different *flow stages*:

- `SYN`: Flows with only the `SYN` message(s), eventually retransmitted by the client multiple times; This is the most common case on darknets, but it happens also on the blocked ports of other deployments or when a responder is unable to cope with the workload;
- `2WH`: Incomplete three-way handshake, where the client ignores (or resets) the `SYN/ACK` message, as in the case of stealth-`SYN` port scans;
- `3WH`: Client and server complete the TCP three-way handshake, but exchange no payload – this is expected in L4-Responders and DPIPot when clients wait

Table 5.2: Deployments and service categories with their basic characterization. All numbers refer to traffic of a /29 network during a full month. For direct comparison, we report numbers only for the first /29 used as darknet. Numbers marked in bold represent clear anomalies.

Deployment	Category	Flows	Flows with L7 Payload	Dest. Ports	Sender Addr.
DPIPot	*All*	**1214 M**[1]	**683 M**[1]	**49 864**[2]	75 k
L7-Responders	*All*	17 M	12 M	**64 919**[2]	94 k
	Database	3 M	166 k	65 535	70 k
	Fileserver	5 M	2 M	65 535	68 k
	Mail	3 M	51 k	65 535	69 k
	Proxy	3 M	24 k	65 535	70 k
	Remote Desktop	8 M	4 M	65 535	70 k
	Terminal	6 M	3 M	65 535	82 k
	Web	3 M	39 k	65 535	73 k
L4-Responders	*All*	8 M	3 M	**49 777**[2]	87 k
	Database	3 M	294 k	65 535	71 k
	Fileserver	3 M	742 k	65 535	69 k
	Mail	3 M	24 k	65 535	71 k
	Proxy	3 M	22 k	65,535	70 k
	Remote Desktop	3 M	268 k	65 535	90 k
	Terminal	4 M	292 k	65 535	71 k
	Web	3 M	34 k	65 535	77 k
Darknet	*None*	2 M	0	65 535	63 k

[1,2] Anomalies discussed in the following.

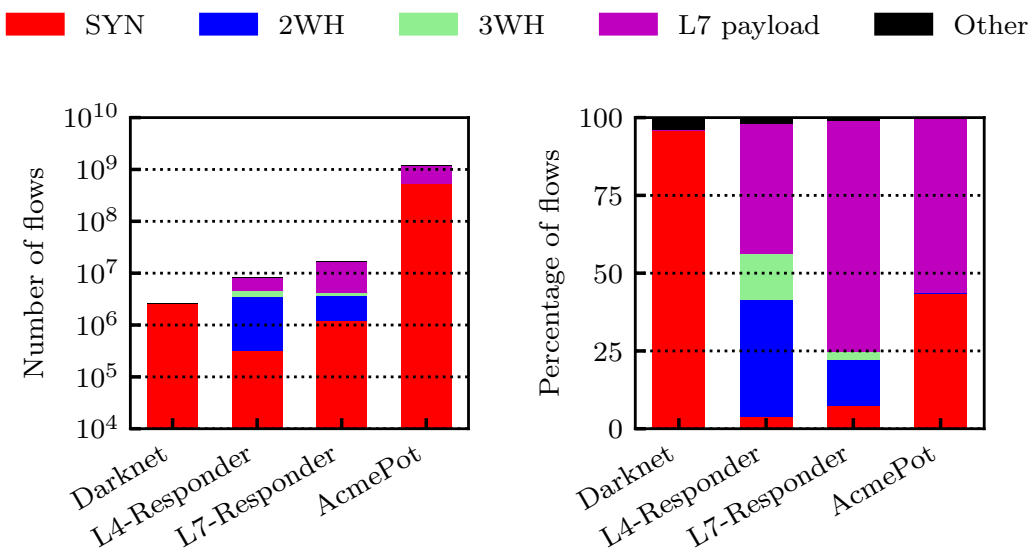
for servers to initiate the conversation;

- `L7 payload`: Client and server open the TCP connection and exchange some application-layer messages.

In addition, I also record malformed TCP messages, e.g., `SYN/ACK` likely arriving due to backscattering or other packets with bogus TCP flags, as well as any other protocol (UDP, ICMP, etc). These cases are however not discussed in this chapter, as they represent only a negligible part of the overall traffic. Here, I analyze data captured over one month, from the 15^{th} of April to the 15^{th} of May 2021. In total I collected about 115 GB of traffic, corresponding to more than 1 300 millions flows coming from more than 600 000 unique IP addresses.

5.4 Macroscopic changes in traffic

In this section I report a high-level characterization of the different deployments to understand how much extra information one would get when starting to reply to unsolicited traffic. For the sake of comparability, I here restrict our analysis to 8 addresses per deployment, focusing on those answering to the *all* category.



(a) Flows (notice the y-log scale)

(b) Percentage

Figure 5.2: Flows reaching different deployments.

5.4.1 Breakdown per flow stage

Fig. 5.2 reports the number of traffic flows received in each deployment, breaking it down per flow stage. The left plot details the number of flows (notice the y-log scale) while right plot details the share in each deployment.

The darknet observes a large majority of TCP `SYN` messages, with a few UDP and bogus TCP segments (about 8% of the total). As soon as the L4-Responders starts replying, the number of flows grows by a factor of 4 compared to the darknet (cfr. Table 5.2). The sudden rise in the number of connection requests may generate a short-term congestion on the L4-Responders, although the L4-Responders shall always perform the full 3-way handshake. Thus, some flows remain in the `SYN` stage. Their share is marginal as visible in the percentages in the right plot. Interestingly, 35% of the flows terminate at the `2WH` stage, most likely corresponding to "host discovery" scans with crafted `SYN` messages (e.g., the TCP SYN Ping performed by Nmap[5]. About one forth of the open TCP flows carries no payload, i.e., likely host discovery actions performed with TCP-connect scan[6].

Moving to the L7-Responders, the number of flows doubles again. The `SYN`

[5] https://nmap.org/man/it/man-host-discovery.html

[6] https://nmap.org/book/scan-methods-connect-scan.html

stage flows are now about 7%. Part of this traffic is again caused by the limits I impose on our infrastructure. However, as I will zoom in later in Sec. 5.6, once I answer traffic in some ports, more scans are observed in other ports too. This effect increases the number of `SYN`-stage flows. Naturally, I observe a strong increase of `L7 payload` flows, which are now about 75% of the total.

Finally, moving to DPIPot, it attracts 3 and 2 orders of magnitude more flows than the darknet and the L7-Responders, respectively. The number of flows grows to billions – about 70 times more than in the L7-Responders, and 600 times more than in the darknet. Here I see around 40% of cases finishing on `SYN` stage, which correspond to periods in which DPIPot hit the rate limiter. That is, the limits I impose on the infrastructure capacity play a crucial role, likely reducing the traffic attracted by DPIPot.

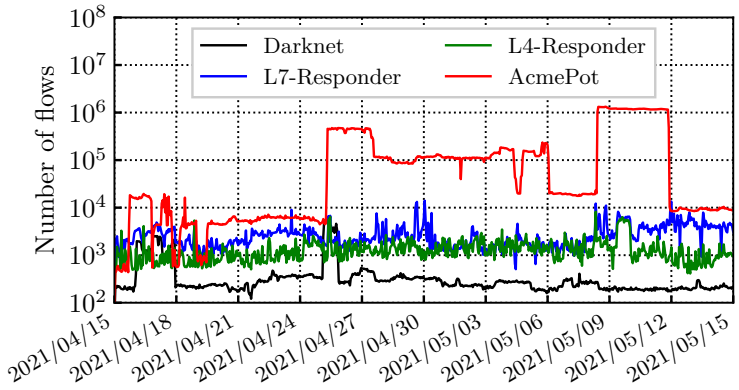
It is worth commenting that the share of `Other` traffic remains similar in all deployments. This suggests that answering TCP traffic does not stimulate senders to generate packets using UDP/ICMP.

5.4.2 Temporal evolution

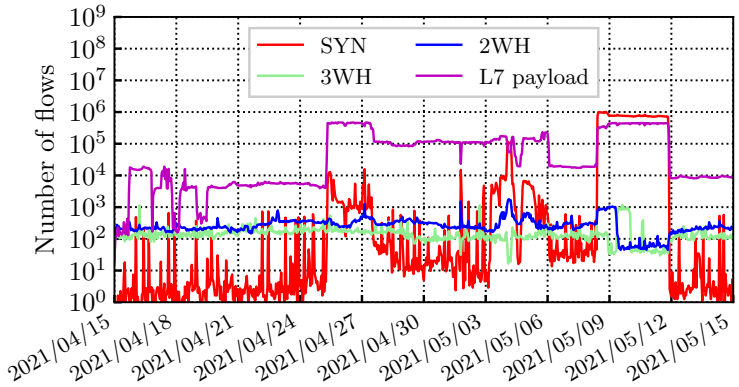
Darknets and honeypots are known to receive variable numbers of flows over time. Our setup is not different: Fig. 5.3a reports the average per-hour number of flows received by each deployment (*All* category again). Notice again the y-log scale. As expected, the darknet is steadily the least contacted deployment with a few hundreds of flows per hour on average, except during sporadic massive scans hitting the address space [22, 79, 137]. Both L4-Responders and L7-Responders show a noisier pattern over time, again with small episodes of increases, sometimes affecting multiple deployments simultaneously (as in April 25th), sometimes being uncorrelated (as in May 7th).

DPIPot registers much more variable figures. For instance, flows per hour top to more than 1 million on May 7th to suddenly vanish on May 12th. I will detail this case in Sec. 5.6.3. As said above, these episodes bring DPIPot to the limits I impose on the infrastructure. In Fig. 5.3b I break down the hourly number of DPIPot flows according to the flow stages. I see for example that the number of `SYN` stage flows is usually negligible. Yet, it increases together with the overall traffic. This is evident during the sudden growth in the May 7th – May 12th interval, where we see a plateau of around 350 000 flows per hour per IP address for `L7 payload` stage.

This explains the first anomaly in Table 5.2: DPIPot manages to answer only 683 million flows out of potentially 1 214 million flow requests.



(a) Darknet, L4-Responders-All, L7-Responders-All, DPIPot



(b) DPIPot flow stage breakdown

Figure 5.3: Temporal evolution of the number of flows.

5.5 Ports, senders and neighbors

In the previous section, I commented on the effects of answering darknet traffic in terms of traffic volumes. I now assess changes on traffic patterns along two axes: the targeted services (i.e., probed ports) and traffic sources (i.e., IP addresses of senders). This section answers the second question, how the presence of active services changes ports and senders, and whether these deployments affect neighbouring darknet addresses. For this purpose, I drop the temporal dimension and analyze the entire aggregate of one-month of data.

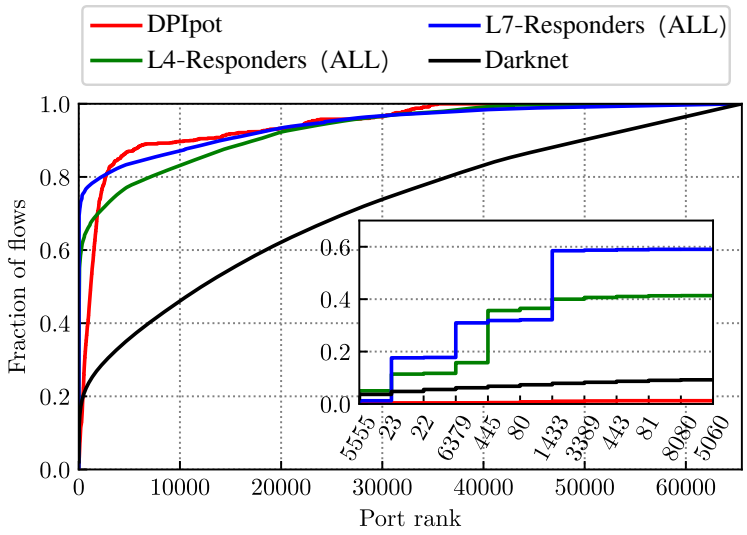


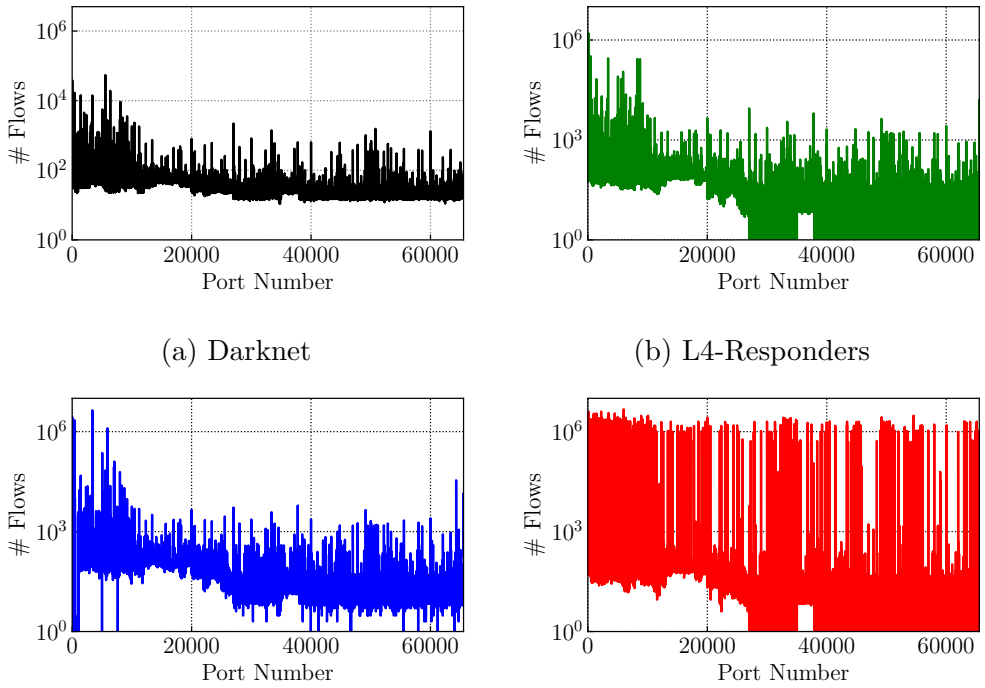
Figure 5.4: Flow distribution per destination port. Ports are ranked according to the received traffic volume in the darknet. Zoom on top-12 ports.

5.5.1 Changes on probed ports

The traffic volume is not evenly distributed over the exposed ports. Already in the darknet, well-known ports are heavily requested while the rest of the ports receive much less attention, likely part of horizontal scans. I assess how such a distribution changes with the increase in the interactivity levels.

Fig. 5.4 reports the cumulative fraction of flows directed to each port, ranked by port popularity (in terms of traffic volume) in the darknet. Focus on the darknet (black curve). A handful ports receive 20% of flows. Then, the share grows almost linearly to cover the whole port range. This plot highlights how senders spend most of the traffic to perform horizontal scans against darknets, i.e., doing host-discovery.

When the deployments start answering requests, the picture drastically changes. In L4-Responders, the top ports account for more than 60% of the flows. This percentage grows to more than 70% in L7-Responders. That is, once a target is discovered, senders activate the next stages of scans or attacks. This is clearly visible in the inset in Fig. 5.4, which details the share of flows in the top-12 ports. For instance, observe the increase at ports 23 (Telnet), 445 (SMB) and 3389 (Remote Desktop). These ports attract way more traffic once opened (L4-Responders), and even more if they expose actual services (L7-Responders, where I confirm that most activity is related to password guessing and brute-force login attempts). Surprisingly, the pattern is not repeated for port 5555, which is often searched due to Android Debug Bridge services. This is the most popular port in the darknet,



(a) Darknet (b) L4-Responders

(c) L7-Responders (d) DPIPot

Figure 5.5: Number of flows per destination port.

and the L7-Responders offer a honeypot on the port [4]. Apparently, either the honeypot does not behave as expected by senders, or the service is not currently receiving attention.

Moving to DPIPot, I register some other differences. Some hundreds of ports get most of the flows, but the remaining ports also gets some uneven distribution of traffic. Unlike the darknets and L7-Responders, more than 15 000 ports never received any flows in the one-month time period, for both DPIPot and L4-Responders. Recall that, for both cases, all ports result open during port scans. *This behaviour corresponds to the second type of anomalies reported in Table 5.2. I conjecture that either senders get busy performing other activities on the already found open ports, or they are more cautious and abort (or time out) scans after finding a high number of open ports.*

Fig. 5.5 clearly depicts this behaviour. For each deployment, I report the number of flows per port, ordered by port number. L4-Responders and DPIPot miss some ports starting from port 27000 (number of flows goes below 1 in the y-log scale). Curiously, notice the continuous group of ports [35000 : 38000] where senders again check all ports. For the sake of completeness, note that few ports go unchecked

also in L7-Responders.

Fig. 5.5 illustrates also the frequency in which ports are visited. As expected, senders usually concentrate their interest on well-known ports below 1024 for darknet, L4-Responders and L7-Responders. On the contrary, DPIPot attracts way more flows on very uncommon ports. Investigating the `L7 payload`, these flows carry Remote Desktop Protocol (RDP), hinting for a specific attack. I analyze this case in details in Sec. 5.6.

5.5.2 Changes on traffic senders

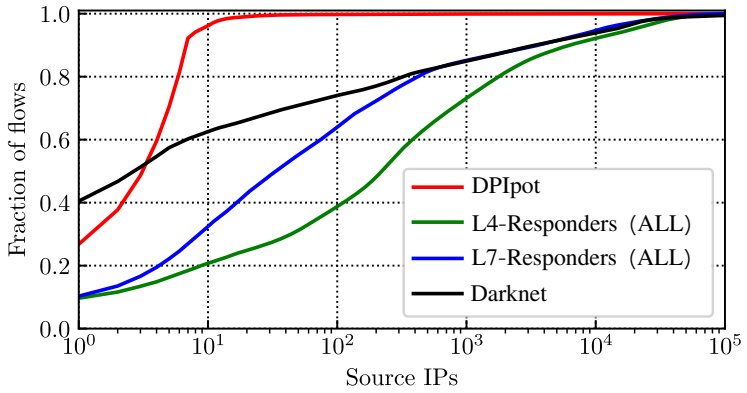


Figure 5.6: Fraction of flows per sender IP address.

I here investigate changes seen in the set of senders contacting the deployments. Fig. 5.6 reports the total fraction of flows from each sender IP. The x-axis reports with a log scale each sender IP ranked according to the volume, for each deployment.

In the darknet, the three most active senders generate 40% of flows. These are well know scanners reported multiple times in blocklists. The top-10 most active responsible of 63% of total flows. Some of them are sending some hundreds probes per hour for the whole time. Some come, probe at high rate (hitting 20 000 flows per hour) and disappear after completing their job. These are also reported multiple times. At last, I observe a tail of more than 63 000 IP addresses. This is in line with previous work [22] that shows darknet traffic is dominated by misconfigured hosts and victim of attacks producing backscattering, with only a minority of senders actually performing scans and attacks. Comparing to L4-Responders and L7-Responders, I observe that flows are more distributed across senders, with the top-10 of them accounting for 20% and 32% of traffic only. Still, the total number of IP addresses is comparable, reaching 87 000 and 94 000. Looking at the tail of the distribution, I observe a lot of senders which exchange few flows with our deployments.

The figure is completely different in DPIPot where the top-10 most active senders account for 95% of all flows. By checking, these senders are involved in the RDP abuse observed on non-standard ports. They generate millions of flows per hour, triggering the rate limiting countermeasures we implemented in DPIPot (cfr. Fig. 5.3b).

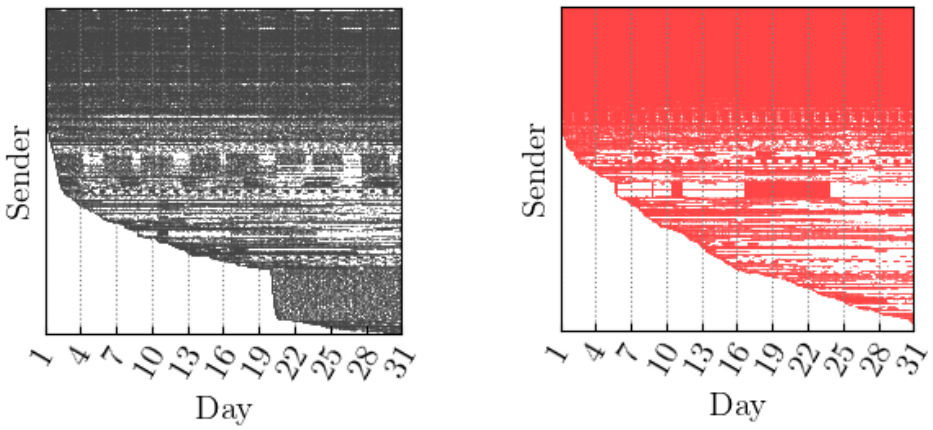
Fig. 5.7 offers a visual representation of the activity of the top-1000 most active senders over time. Each row corresponds to a single IP address. A dot is present if that IP address is active in that hour. I register the presence of senders that are active most of the time, and the continuous arrival of new senders. For instance, a group of 200 new senders appears on day 20 in the darknet. Such bulk arrivals are likely due to bots that perform a coordinated scan reaching our address space. In general, different patterns are visible: some senders are persistent, while others keep coming back periodically. A few interact only for some time before disappearing and never coming back. The latter is more visible in Fig. 5.7b for DPIPot rather than the darknet in Fig. 5.7a This sudden and bursty activity patterns are typical of being part of botnets that perform their scan and attacks to then move on the following victims.

Next, I compute the Jaccard indexes to check if senders contacting different deployments are the same. Given two deployments, I extract the set of IP addresses seen in each of them for the whole period, and compute the ratio between the intersection of the sets over the union. Table 5.3 details the results, where I further distinguish between the darknets in the *same* /24 address space as the deployments, and the *other* /24 network. The latter distinction allows us to answer the question of whether the presence of responders impacts the neighboring addresses.

It turns out that activating responders in the same /24 address space "pollutes" the darknet addresses, which get contacted more frequently by the same senders that reach our responders. Indeed, the Jaccard index shows that 34-43% of IP addresses are in common between the responders and the darknet sharing the same address space. Conversely, less than 15% of senders seen in the *other* darknet are also seen in the responders. This suggests that senders involved in the initial scan phase are not the same senders involved in the subsequent phases. Activating responders thus increases the visibility on senders. The low value of the Jaccard index is due to occasional senders hitting our address space - see the tail in Fig. 5.6 - due to backscattering, misconfigurations, and bugs [22].

Table 5.3: Jaccard Index of senders hitting different deployments.

	Dark/same	L4-Resp	L7-Resp	DPIPot
Dark/other	*0.128*	*0.142*	*0.137*	*0.143*
Dark/same	–	0.366	0.341	0.420
L4-Resp	–	–	0.429	0.396
L7-Resp	–	–	–	0.372



(a) Darknet (b) DPIPot

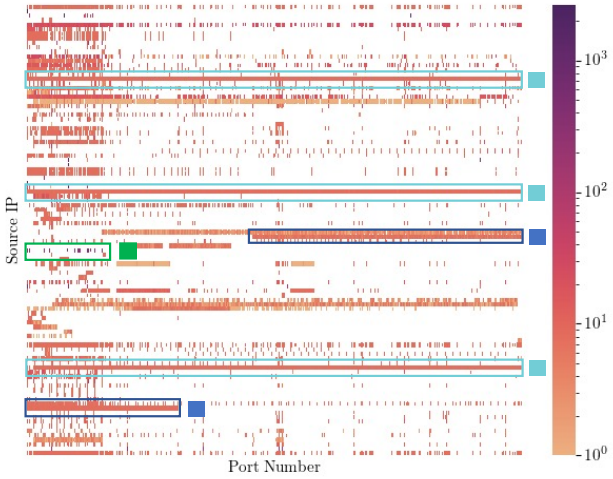
Figure 5.7: Activity pattern of top-1000 sender IP addresses. Each row corresponds to a sender IP address.

5.5.3 Who does what?

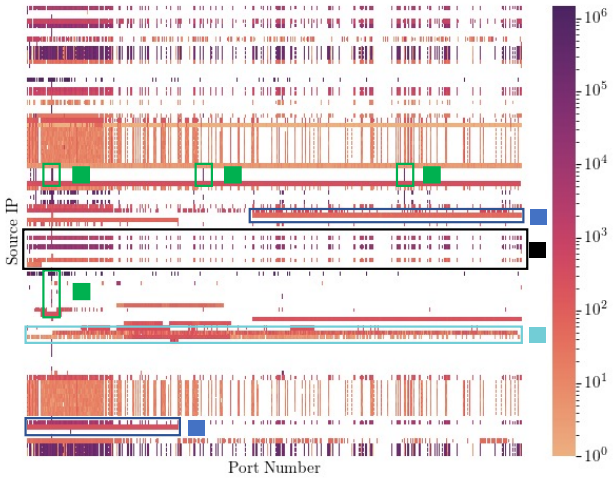
At last, given the diversity of the destination ports and the sender IP addresses, I quickly investigate what attack or scan patterns the top-100 most active senders perform in each deployment. For the sake of brevity, I report only the simplest and the most complex and diverse scenario of the darknet and the DPIPot, but similar activities are visible on L4-Responders and L7-Responders as well. Fig. 5.8 plots the activity pattern. The x-axis reports the destination port of each flow, while each row refers to a single sender, ordered by its IP address, i.e., putting nearby addresses starting with the same initial octet. A dot is present if that IP address sends a flow to such port. The darker the color, the larger the amount.

Check first Fig. 5.8a for the darknet. Clear patterns emerge - some I highlight using colors. First, some few horizontal scans are visible (cyan), with senders checking all ports. Second, I observe some vertical scanners (green) - i.e., addresses that target only few ports. Third, some senders target a large subset of ports, stopping the scan after some thousands of probes (dark blue).

Moving to Fig. 5.8b for DPIPot, some clearer patterns emerge. Besides the horizontal scanners (cyan) (which however do not probe all ports anymore - cfr. Fig. 5.5d), very targeted scans are visible on very few ports with up to thousands of flows (green). These are vertical attacks that target well-known ports and services. Not shown here, they are present in L7-Responders too. A large number of coordinated scanners emerge too, i.e., senders that target the same few thousands ports (most of which in the [1:1024] range), and fewer selected ports higher than 1024 (black). Most of these are senders involved in the RDP attack on non standard ports. At last, some smaller group of scanners appear (dark blue); each scanner



(a) Darknet



(b) DPIPot

Figure 5.8: Flows from top-100 sender to destination port. Addresses are sorted numerically.

probes a different subset of ports, splitting the port range in a coordinated effort.

5.6 Gain in service-specific deployments

Here, I focus on the extra information L4-Responders, L7-Responders and DPIPot offer compared to darknets. By evaluating the traffic gain of the several

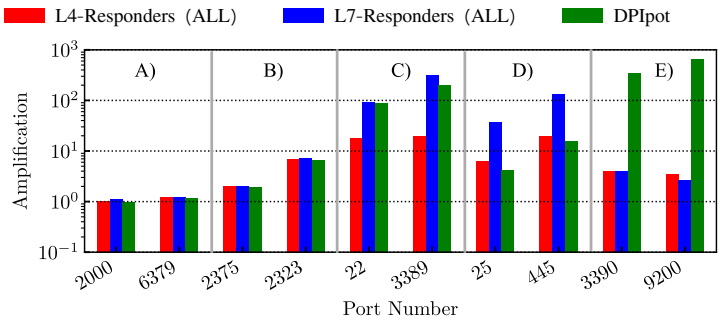


Figure 5.9: Gain for most targeted ports.

deployments, I answer our last two questions, i.e., whether the presence of specific services attracts traffic to other services, and what happens when one starts answering to non-standard ports.

5.6.1 Gain in service requests

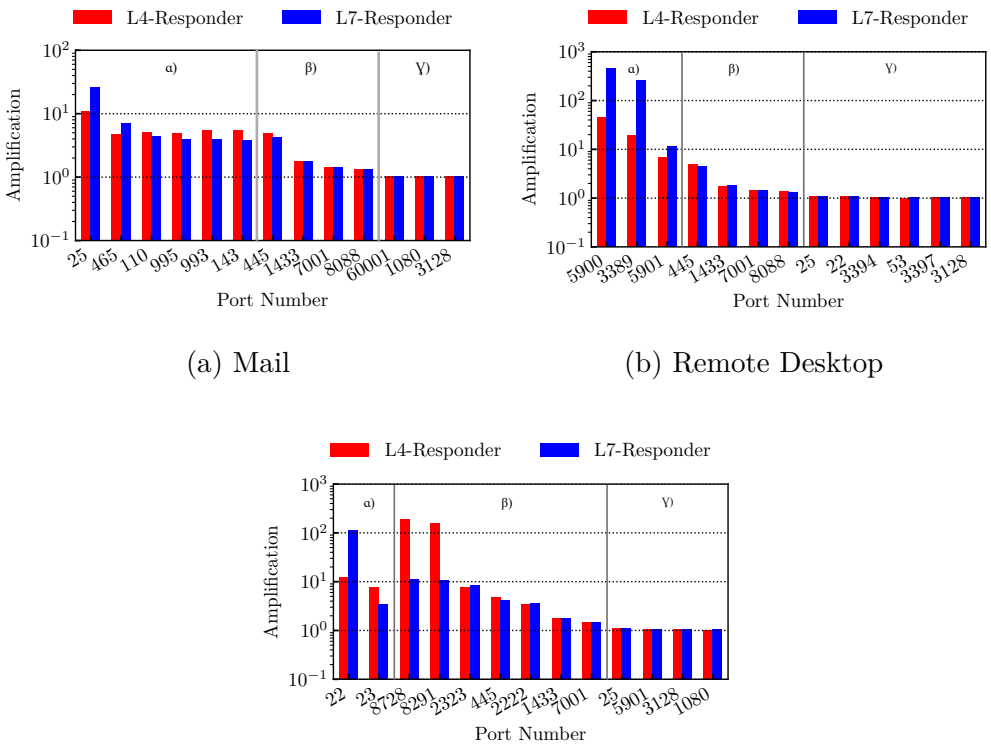
To quantify the extra traffic per deployment, I define the *Gain* as the ratio between the number of flows seen on the 8 IP addresses of a specific port(s), and the number of flows directed to the same port(s) on the 8 IP addresses belonging to the darknet. The aim of this quantity is to define the amount of additional data gained by active probes compared to non-active ones.

First, I run a preliminary test to verify whether the gain changes when comparing IP addresses belonging to the same deployment. For this, I take all /29 darknets included in the single non-polluted /24 address space, and compute – for each destination port – the gain for each darknet pair. I verify that the distribution of the pairwise gains is centered between 0.9 and 1.1. I therefore consider significant any gain outside this range.

Fig. 5.9 shows the gain for some selected ports. I identify five major categories of behavior, which are labeled with capital letters. I provide two examples per category:

A) Invariant (around 50 000 ports): the traffic reaching these ports does not change significantly from the darknet to the other deployments. Ports like 2 000 and 6 379 receive only *port scan* attempts, whose volume does not change when responders are present;

B) Homogeneous (around 13 000 ports): senders find possible services on some open ports. These open ports trigger senders to contact several other ports on the host – e.g., ports 2 375 and 2 323 in the figure. However, for these



(a) Mail

(b) Remote Desktop

(c) Terminal

Figure 5.10: Gain for selected deployments. β marks cases of *Side-Scans*.

ports, senders do not send any L7 payload - e.g., because waiting for servers to initiate the exchange. Here, L7-Responders and DPIPot behave just like L4-Responders;

C) L7 client-initiated (around 500 ports): these are clear cases of open services on default ports with client-initiated protocols, e.g., SSH and RDP on ports 22 and 3 389. Both L7-Responders and DPIPot are effective to engage with the senders. Since frequent attacks are present, we observe very large gains. L4-Responders are less interesting for the senders, with reduced gain;

D) L7 server-initiated (around 10 ports): open services on default ports for which the senders expect the server to initiate the L7 exchange. In this case, the L7-Responders vertical honeypots are more effective, while DPIPot behaves as the L4-Responders, e.g., on SMTP and SMB on ports 25 and 445, respectively;

E) Large-scale attacks on non-standard ports (around 1 500 ports): Senders discover particular services on non-standard ports and perform large attacks.

Table 5.4: Traffic gain for L4-Responders and L7-Responders. Cases in which no gain is observed is marked with a hyphen.

L4-Responders								
	DB	*File*	*Mail*	*Proxy*	*RD*	*Terminal*	*Web*	*Others*
DB	**15.4**	4.3	–	–	–	–	–	–
File	1.6	**42.0**	–	–	–	–	–	–
Mail	1.5	4.1	**6.5**	–	–	–	–	–
Proxy	1.5	4.2	–	**2.7**	–	1.2	–	1.2
RD	1.5	4.2	–	–	**21.2**	1.6	–	–
Terminal	1.5	4.1	–	–	–	**9.3**	–	1.3
Web	1.5	4.2	1.4	1.2	–	1.3	**8.1**	–

L7-Responders								
	DB	*File*	*Mail*	*Proxy*	*RD*	*Terminal*	*Web*	*Others*
DB	**9.3**	3.9	–	–	–	–	–	–
File	1.6	**116.3**	–	–	–	–	–	–
Mail	1.5	3.6	**9.6**	–	–	–	–	–
Proxy	1.5	3.8	–	**2.8**	–	–	–	1.2
RD	1.5	3.8	–	–	**254.9**	–	–	–
Terminal	1.5	3.6	–	–	–	**46.6**	–	1.2
Web	1.5	3.8	1.2	1.2	–	1.2	**5.3**	–

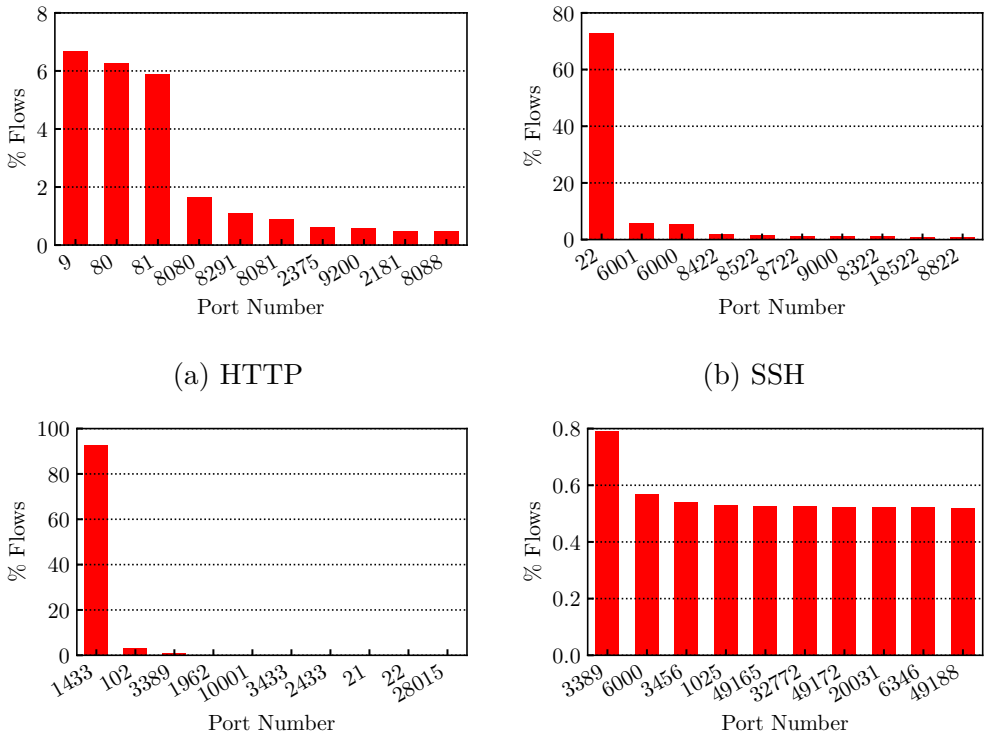
Only DPIPot, which identifies the L7-protocol, spots such behavior. In particular I have witnessed an extensive RDP attack on multiple non-standard ports, resulting in around 1 500 ports for which DPIPot gain grows to almost 1,000.

5.6.2 Targeted services and *Side-Scans*

So far I focused on the responders that support *All* services at the same time. I now check what happens if I partition responders so that they behave as vertical services. For this, I consider L4-Responders and L7-Responders, with categories/ports/applications defined in Table 5.2. For each category, we compute the gain with respect to the corresponding categories/ports/applications in darknet addresses.

Table 5.4 summarizes results. For each vertical deployment, we report the gain only when significant. Rows report the category of the deployment while columns report the corresponding categories in the darknet as reference. As expected, activating specific services attracts the attention of senders on them (see main diagonal, in bold). L4-Responders suffice to observe more traffic, but L7-Responders clearly generate much more interaction. Exceptionally, L4-Responders see higher gain than L7-Responders in some cases (e.g., *DB*). This is likely a consequence of our lack of honeypots for some ports of these categories in the L7 backends (see Table 5.1). In this case, while L7-Responders reply with an uninteresting response (reset the connections), the limitation to TCP handshake offered by L4-Responders further engages the senders.

Surprisingly, I observe also significant gains on services for which the deployment



(a) HTTP

(b) SSH

(c) MsSQL-TDS

(d) RDP

Figure 5.11: Flows percentages on top-10 ports for DPIPot and different L7 protocols.

does *not* answer, i.e., where it drops the `SYN` packets. Regardless of the deployment, once senders find an IP address that is alive (i.e., answering a popular service), they target other ports in the *DB* and *File* categories. The case of the *Web* category is particularly interesting: When a service is found active on ports typically hosting HTTP services, senders apparently start targeting multiple other services/ports on the same host. I refer to this phenomenon as *Side-Scan* activity.

Fig. 5.10 reports some of the most relevant *Side-Scan* examples. α) marks the well-known (and open) ports for the category. Here, I get as expected significant gains, with L7-Responders getting significantly more traffic than L4-Responders for some honeypots. β) marks those *Side-Scan* ports that suddenly get targeted - despite being blocked for the particular deployment. These are the ports senders target in vertical attacks/scans triggered by a different category. Finally, γ) exemplifies some ports that remain invariant, i.e., they are neither the initial target, nor reached in *Side-Scans*. Most ports fall in this class.

These cases clearly show benefits of activating some services on darknets. The

vertical deployments allows the discovery of ongoing *Side-Scan* for services, which otherwise would have remained unnoticed in the darknet. When opening ports of the *Mail* category (plot in the left-hand side), I observe significant gains on ports (445, 1 433), which are usually used in *File* and *DB* services. I record *Side-Scans* also on ports (7 001,8 088). Similar effects can be seen for the Remote Desktop category.

Less unexpected, ports (2 222, 2 323) are often used as alternative ports for terminal services. Senders *Side-Scan* these ports when finding standard terminal ports open (see Fig. 5.10c). Ports (8 728, 8 291) are known to be vulnerable services in old versions of software routers. In particular, I observe frequent "port-knocking" attempts: the sender checks for port 22 first; if open but no banner is offered, it checks ports (8 728, 8 291). L7-Responders do offer a banner on port 22. Thus, flows on ports (8 728, 8 291) are smaller than in L4-Responders that offers no banner on port 22 [124].

5.6.3 DPIPot additional visibility

Table 5.5: Top-5 protocols recognized in DPIPot.

Protocol	Flows	Sender Addr.	Dest. Ports	% of Flows on Standard Ports
RDP	329 652 678	1 415	28 333	0.8
HTTP	444 715	13 705	9 381	6.2
TLS	221 565	2 806	11 999	4.6
SSH	119 698	1 097	187	72.9
MsSQL-TDS	31 596	3 193	448	92.6

Eventually, I dig into DPIPot data to see if its ability to reply to L7 requests on any port increases visibility and originates significant changes.

Table 5.5 reports a summary of the most common application protocols detected by DPIPot. I observe a vast majority of RDP flows - with 1 415 senders generating more than 330 M flows in one month. These senders target more than 28 thousand ports, with the standard port 3 389 accounting for only 0.8% of flows. That is, the majority of this attack would go unnoticed on darknets. I already recorded this behaviour in Fig. 5.5 and Fig. 5.6 where the IP addresses involved in this attack dominate the traffic DPIPot collects. DPIPot spots also other popular protocols like HTTP, TLS and SSH, where thousands of senders target thousands of ports. Some of these attacks focus mostly on the default port - like SSH or MsSQL-TDS where 72.9% and 92.6% of the flows goes to the default ports.

To check how senders choose the port to probe for a given protocol, Figure 5.11 details the most popular ports target of some L7 protocols. Start from the HTTP case. Port 9 results as the most popular port to receive HTTP requests. This is a *Side-Scan* performed by an Internet mapping project of the University of Michigan,

which targets port 9 (about 30 000 flows) and 7 (about 50 flows only), sending bogus HTTP requests [103]. This scan activity would likely go unnoticed on traditional honeypots. Besides this curious scan, DPIPot recognizes and correctly handles HTTP requests on non-standard ports. Given the popularity of solutions based on HTTP protocol, it is not surprising to see senders probe open ports with HTTP requests.

Move to SSH now. Here, most flows target port 22. Yet, the senders do check other ports where system administrators may move the SSH service, e.g., by adding some digits to the 22 ports as in (8 422, 8 522, 18 522). This behavior suggests a targeted *Side-Scan* where senders generate the port to target with some domain-knowledge driven algorithm.

The *Side-Scan* using the MsSQL-TDS protocol is even more vertical. Most of the attacks are directed to the default port 1 443, but some few requests go to port 102, likely trying to abuse some Microsoft Exchange service.

At last, the RDP case is worth more details. RDP has become a viable solution for malicious hosts for installing ransomware [153] via attacks that start with password brute-force [26], or exploit a well-known backdoor [13].

Thanks to DPIPot, I observe 1 415 senders performing a password brute-force attack. The attackers however execute the brute-force in almost any port announcing RDP support. Figure 5.12 shows the targeted ports, ranked per number of received flows. Notice the log-log scale. The step-wise behavior of the figure suggests the presence of a group of 1 000 ports that gets most requests, followed by a second group of ports which are contacted less frequently. This second group may be due to an initial discovery horizontal scan, after which senders come back to perform the brute force password attack. The inner plot shows that there is also a clear pattern for the top-300 ports. Checking the behavior of each sender, I recognize three macro-categories:

- Senders (around 700) that vertically probe only standard RDP port 3 389 and the immediately adjacent ones;
- Senders that focus on a small group of selected ports (e.g., ports 1 289, 23 390, 1 025, 3 418, 50 000, 554, 3 336) - likely chosen via domain knowledge. The four IP addresses involved in this attack belong to the same network and have never been reported at the time of writing. They generate 3.5 million flows;
- Senders that scan thousand of ports (16 IP addresses). These addresses have been reported as heavy scanners [150] and perform very similar activity. This suggests they are part of the same botnet.

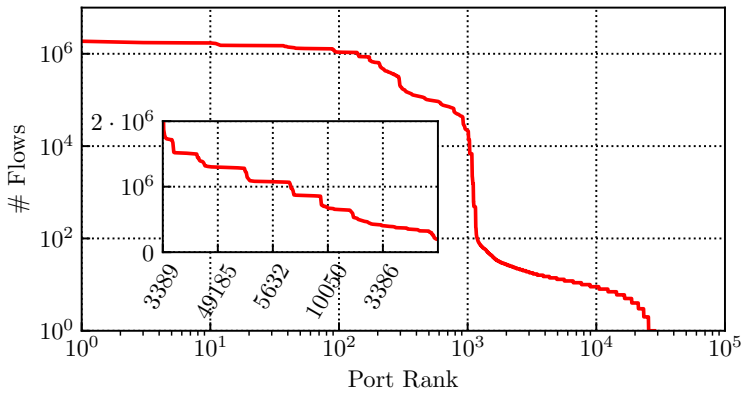


Figure 5.12: Number of flows per port for RDP traffic. Zoom on first 300 ports in inner axis.

5.7 Conclusion

In this chapter I analyzed the impact of deploying more and more interactive responders on the darknet address space. The obtained results show the potential of systematically engaging with senders. I confirm some already known patterns, such as the expected increase in traffic when active services are deployed on the darknet: Indeed, the number of flows grows by orders of magnitude with increasingly interactive responders. Vertical honeypots attract 10 times more flows than darknets. DPIPot pushes this increase to 600 times, thanks to its ability to answer application traffic on non-standard ports. This growth creates temporal bursts of traffic that challenge the deployment itself and call for protection mechanisms to avoid collapsing the infrastructure and biasing the collected data.

I further observe that answering to some ports engages senders, which activate the next stage in their scans or attacks. This increases the traffic and sometimes challenges the monitoring infrastructure. Answering all ports traps senders in some activities, possibly limiting/biasing their activity. Moreover, most of traffic comes from few thousands senders that are involved either with vertical or horizontal scans and attacks. IP addresses involved in darknet scans are typically different from those seen in the later attack stages. And backscattering generates occasional senders that pollute senders sets. Unlike vertical honeypots, DPIPot allows the observation of new scan patters where also non-standard ports get the attention of attackers.

I also shed lights on new events, such as the *Side-Scans* attracted by opening different services both on standards and non-standard ports. Such patterns would remain otherwise unnoticed on darknets or classic honeypots alone, while it is clearly highlighted by DPIPot.

I show that each deployment has its own benefits, unveiling different activities and bringing new perspectives. Combining the several interaction levels augments visibility. However, deployments may impact each other (e.g., polluting neighboring addresses) and may foster traffic increase to the point of saturating the monitoring infrastructure.

Beyond these findings, several challenges are waiting ahead of such hybrid infrastructures. For example, the large amount of collected information calls for automatic methods for analyzing the data, uncovering correlation between deployments, fingerprinting senders and, ultimately, identifying the rise of novel scans and cyberthreats.

Chapter 6

Sensing the Noise: Uncovering Communities in Unsolicited Traffic

The work I present in this chapter is mostly taken from my paper *Sensing the Noise: Uncovering Communities in Darknet Traffic*, presented at the 18th Mediterranean Communication and Computer Networking Conference (MedComNet 2020) [135].

6.1 Introduction

As highlighted in the previous chapters, understanding the underlying phenomena behind unsolicited traffic is challenging given the large amount of packets and the heterogeneity of events generating them. Being darknets and honeypots a continuous targets for the most diverse types of scans, a manual exploration of the collected traces is not possible, nor efficient. The analysis of such traffic requires methods to separate occasional (and likely unimportant) events from large-scale coordinated actions, e.g., due to botnets. Several works propose methods to categorize darknet traffic [2, 57, 47, 79, 44], but they usually rely on domain knowledge to create static categories of traffic (e.g., misconfiguration, backscattering, etc). As such, they may miss events that diverge from expected signatures. More important, there is a lack for automatic ways to uncover correlations among different events, which could help reducing the manual work when interpreting unsolicited traffic.

When processing unsolicited traffic, grouping packets from different sources may suggest the presence of coordinated activities, due for example to the propagation of a botnet infection over different hosts. Such coordinated actions may result in temporal or spatial correlations, e.g., apparently unrelated sources cooperating to scan for a particular service, and they are invisible if hosts or packets are analyzed

individually. Data should therefore be represented in an informative way, so to ease the inspection and uncovering of hidden patterns.

This chapter investigates whether graph mining techniques can help to uncover such macroscopic coordinated events in unsolicited traffic. I represent traffic as a bipartite graph linking traffic sources to the contacted destination ports. I then run community detection algorithms over such graphs, in the search for devices performing similar activity against the same sources in the same time interval.

I firstly evaluate the methodology using only TCP traffic, collected in two distinct darknets. I tune and test the methodology on three weeks of data captured from a darknet in Italy, composed by 3 /24 IPv4 networks. I then apply the methodology to one day of traffic captured in a /19 darknet deployed in Brazil. I finally validate the methodology on one month of honeypot traffic reaching DPIPot (cfr Sec. 5.6.3). In all cases I found communities performing very homogeneous activity. I discuss the composition of the most relevant communities, their characteristics and peculiarities, showing some relevant behaviors that the tested algorithm is able to automatically detect. In particular, I find (i) communities composed by thousands of sources that focus on popular services; (ii) communities that focus on horizontal scans for vulnerable services.

This work is a first step towards a methodology to automate the analysis of darknet traffic. The results confirm that unsolicited traffic can be automatically characterized by using graph mining techniques to highlight interesting patterns over space and time. In the following, Section 6.2 describes the applied graph mining methodology. Section 6.3 describes the datasets and provides a basic characterization of the data. Sections 6.4 and 6.5 describe the output of the graph mining and community detection analysis on darknets and honeypots, respectively. Section 6.6 concludes the chapter.

6.2 Methodology

In this section I describe the graph definition (Sect. 6.2.1) and the community detection algorithms (Sect. 6.2.2) I apply.

6.2.1 Graph definition

I define a graph $G(V, E)$ with V being its set of nodes and E its set of edges. I want to represent the activity of remote sources sending traffic to my infrastructure. After I take into account different aspects, I pick a definition for $G(V, E)$ that provides us not only good semantics, but also a manageable graph.

$G(V, E)$ is a weighted bipartite graph. Nodes in V represent, on the one hand, the sources sending traffic to darknets and honeypots and, on the other hand, the destination *ports* contacted by the sources. I call S the set of nodes in V

representing the traffic sources and P the set of nodes representing the contacted ports. There is a edge e between a node in S and a node in P if the source has sent packets to the given port. The weight w_e of e is the number of packets observed for the pair of nodes in the given time interval.

I focus on port numbers as they provide an indication of the services searched by sources. The decision on how to represent the sources is however harder, and darknets and honeypots data require to be treated differently. In the darknet case, creating a node for each single sender IP address may result in very large graphs, and therefore lead to scalability issues. Grouping addresses according to networks is instead a more prominent alternative. To avoid setting a completely arbitrary aggregation, here I map the IP addresses to their respective Autonomous Systems (ASes).[1] As such, I report coordination among IP addresses that belong to different ASes. For the DPIPot case, instead, I extract a separate graph for each L7-protocol (see Tab. 5.5). By doing so, I am able to refine the granularity of the source nodes up to the single source IP address, still maintaining the problem tractable.

6.2.2 Detecting communities

Community detection aims at finding subgroups of nodes that are densely connected – i.e., forming communities. In this specific case, communities would represent sets of ASes (IPs) that contact similar sets of destination ports in a given time interval. For instance, a group of remote sources that behave similarly due to a botnet infection, or nodes under the control of the same attacker in distinct source ASes (IPs), that aim at finding vulnerabilities on similar targets.

I here consider the Greedy Modularity Algorithm (GMA) [25].[2] This technique measures how strongly a graph can be separated into modules – i.e., groups of nodes that are strongly connected inside the group, while loosely connected with nodes belonging to other groups. The idea behind the algorithm is that a random graph is not expected to show cluster structures with condensed nodes and edges, while nodes having some sort correlation will form a modular structure. To recognize modular structures, the GMA exploits the concept of modularity [110], defined as:

$$Q = \frac{1}{2m} \sum_{i,j} [w_{ij} - \frac{k_i k_j}{2m}] \delta(c_i, c_j) \tag{6.1}$$

where w_{ij} represents the weight of the edge between node i and node j, k_i is the sum of the weights of the edges on i, c_i is the community to which i is assigned, $\delta(u, v)$ is 1 if $u = v$ and 0 otherwise, and $m = \frac{1}{2} \sum_{i,j} w_{ij}$. The algorithm

[1] https://pypi.org/project/pyasn/

[2] I have also considered the Label Propagation Algorithm [118], which however produces less meaningful results, e.g., putting almost all nodes in a single community.

starts by initializing a community per node; at the first iteration it hence yields $|V|$ communities. The second iteration proceeds by calculating the modularity between each nodes and its neighbours, performing the merge between the pair of nodes that have the highest modularity gain. At each iteration the algorithm merges together the neighbouring communities yielding positive gains in modularity. It stops when it is unable to merge any new elements to any community. As $G(V, E)$ is a bipartite graph, our communities will contain both nodes belonging to S (sources) and P (ports). Every packet (flow) in my datasets, which originate the edges in $G(V, E)$, is labeled with two communities: The community of its source AS (IP) and the community of its destination port. When quantifying activity of communities in terms of packets, I therefore split results into *AS (IP) communities* and *port communities*. In other words, an AS (IP) s_i will belong to a single AS (IP) community c_j (also containing other sources that have performed activity similar to s_i), but its activity will be reflected into more than one port community, implying that not all ports contacted by the s_i are also part of c_j. The same concept applies when tackling the problem from the point of view of the destination ports.

6.3 Darknet datasets

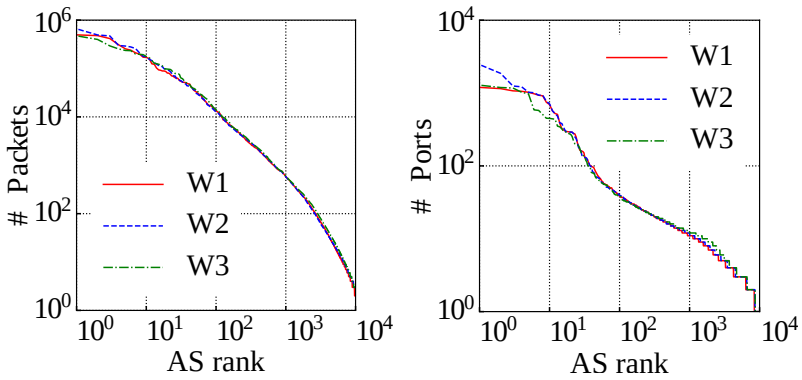
I rely on packets captured from the Italian and Brazilian darknet, respectively. I tune and evaluate the methodology using three weeks of traffic collected on the Italian darknet in January 2020. Afterwards I use the Brazilian traces to confirm my findings (see Chapter **??** for the datasets characterization). Since both darknets are physically located in different continents and logically located in far away IPv4 ranges,[3] I can verify whether the main events observed in a darknet are also observable in another network.

In the following I report some basic characteristics of the darknet traffic. Note that I show only results for the Italian darknet, as they are needed for tuning the methodology. Figures for the Brazilian darknet are qualitatively similar, even if it receives much more packets, since it aggregates more addresses than the Italian one.

6.3.1 Popularity of ASes

Figure 6.1 breaks down the data according to source ASes. Different lines represent each of the three weeks in the data, I analyse every week independently from the others.

[3]Privacy requirements imposed by the network operators prevent me from disclosing the IPv4 prefixes hosting the darknets.



(a) Packets per AS (b) Contacted ports per AS

Figure 6.1: Per-AS breakdown in the Italian darknet. Notice the log-log scales.

Figure 6.1a shows the number of packets received from each AS. ASes are ranked in the x-axis according to the number of packets they send. Results are almost coincident for the three weeks. Most ASes are populated by sources which send only few packets to the darknet in a week (notice the log-log scale). Yet, a small group hosts some *heavy hitters* which send thousands of packets. Notice how the top 1 000 ASes are origin of more than 1 000 packets each, some of them of hundreds of thousands of packets (leftmost part of the plot).

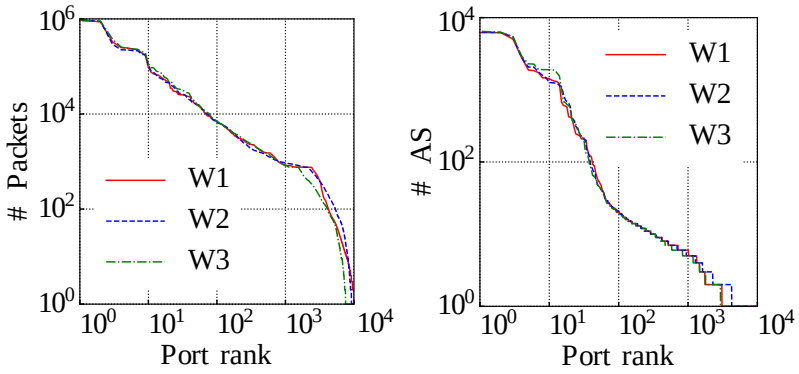
More interesting, Figure 6.1b shows the ranking of ASes according to the number of destination ports their hosts contact. We see that only a small number of ASes targets 100 ports or more (leftmost points). Hosts targeting lots of destination ports are likely performing Internet scans. The remaining ASes target a small number of destination ports, suggesting either targeted activity or random behaviour – e.g., hosts in different ASes collaborating for distributed port scans.

6.3.2 Popularity of ports

Figure 6.2 depicts a breakdown according to destination ports. As for the previous section, I show only results for the Italian darknet.

Figure 6.2a depicts the number of packets received by each port. Again only a small subset of ports sees a significant number of packets. I observe that most of the 65 536 available TCP ports do not receive any packets in a week, whereas most of the targeted ports receive less than 100 packets (notice the log scales). Figure 6.2b depicts the number of unique ASes that contact each port. The leftmost part of the plot shows that only a minor number of ports is contacted by a large number of ASes. Only the most contacted ports receive packets from 30 or more ASes.

The ports receiving lots of packet are the ones corresponding to services often targeted by remote attacks. In Figure 6.3 I highlight the most contacted ports



(a) Packets per Port (b) Source ASes per Port

Figure 6.2: Per-port breakdown in the Italian darknet. Notice the log-log scales.

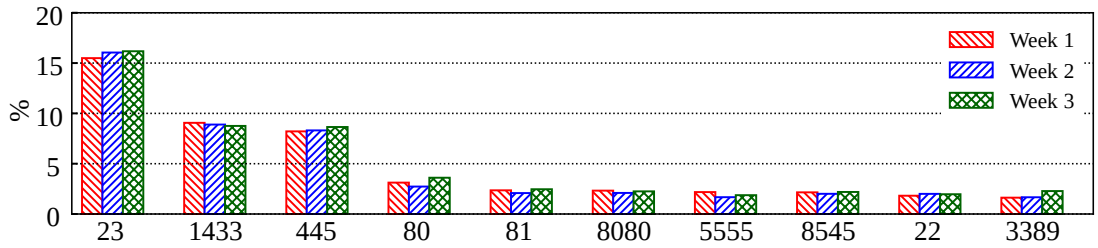


Figure 6.3: Percentage of packets directed to the top-10 destination ports.

in the Italian darknet, along with the percentage of packets received by each of them. I see that ports hosting terminal services (e.g., port 23, 22 and 3389), web servers (e.g., ports 80 and 8080) and databases (e.g., port 1433) dominate the list, attracting large percentage of traffic.

All in all, I record heavy-tails in the popularity of both ASes and ports. For building the graphs in the coming sections, I filter out traffic going to *unpopular ports*. This step is relevant, since some graph mining algorithms do not scale well for large graphs. I set the threshold at 100 packets based on Figure 6.2a. I argue that most packets going to those ports are due to misconfigurations and backscattering [137]. The latter, in particular, are packets sent by *victims* of spoofed IP attacks, which therefore are less likely to show interesting coordination. This threshold allows to filter out (on average) 93% of the ports, while still retaining 98.7% of the observed packets in the traces. Given that my objective is to identify the spread of wider phenomena and possible coordinated actions among ASes, I choose instead not to set a threshold to filter out the sources.

6.4 Darknet communities

In this section I summarize the communities found in both darknets. The resulting graphs are composed by an average of around 16 000 nodes and 67 000 edges per week (Italy), and around 32 000 nodes and 142 000 edges (Brazil).

6.4.1 Community popularity

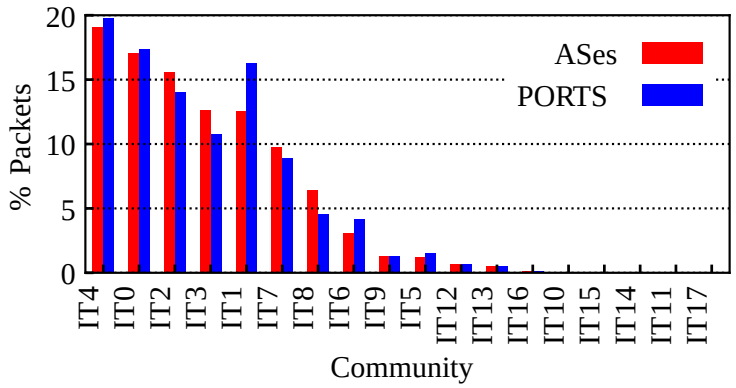
Figure 6.4 provides a high-level view of the communities found for the first analyzed week (day in the BR case). The community detection algorithm has found 18 (20) communities of variable sizes for Italy (Brazil). The figure shows the percentage of packets per community. Recall that each packet may be associated with 2 communities, one for its AS and one for its destination port. The total number of packets per community is however very similar regardless on whether ports or ASes are considered for this association.

The top three communities are involved in more than half of the total weekly traffic in both darknets. Intuitively, these communities may include sources searching for common Internet services, e.g., scans for the popular ports seen in Figure 6.3. I will investigate this hypothesis later. Observe that only the top 12 (13) communities in Italy (Brazil) have a significant number of packets. Communities with negligible number of packets are mostly formed by a few ASes that target a single port. I ignore these negligible communities in the analysis that follows. Tables 6.1 and 6.2 report a more detailed breakdown of each community. The communities are sorted as in Figure 6.3, according to the number of packets they include. Most of the communities target all the available addresses in the darknets. Some of the largest communities prove to collect either more sources (as in IT1 or BR0), or more destinations (as in IT3 and BR3). Only by having a look at these data I am able to identify more specific and less evident events as possible net scans on a single port (e.g., as in IT10, IT17 and BR17), or neglect what really may seem isolated phenomena, as the ones in BR13.

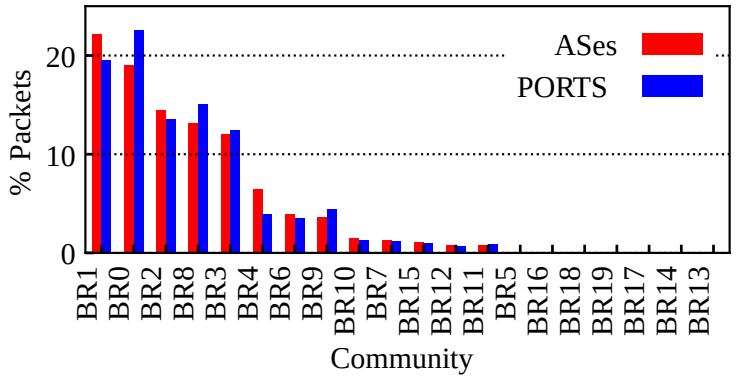
6.4.2 Community structure

Figure 6.5 summarizes the structure of popular communities. It depicts a scatter plot that compares the number of ports and ASes in the communities. Every dot represents a community, and the dot sizes represent the average number of IP addresses per AS belonging to it. It assists in understanding how pervasive the activities are in the ASes, e.g., whether only some few clients per AS participate in suspicious activities, or whether lots of hosts in the ASes join them.

Focusing on the top figure (Italy), see how the communities that group more ASes (i.e., IT1, IT4 and IT0) have a low number of hosts. Moreover, these ASes send packets to a small number of destination ports. These communities are also



(a) Italy - Week 1



(b) Brazil - Day 1

Figure 6.4: Distribution of packets per community.

among the largest ones uncovered by the modularity algorithm. In a nutshell, the results suggest the existence of sparse sources, distributed in a large number of ASes, targeting some specific (popular) ports. This signature matches scans towards specific services, carried on by very distributed sources (e.g., botnets).

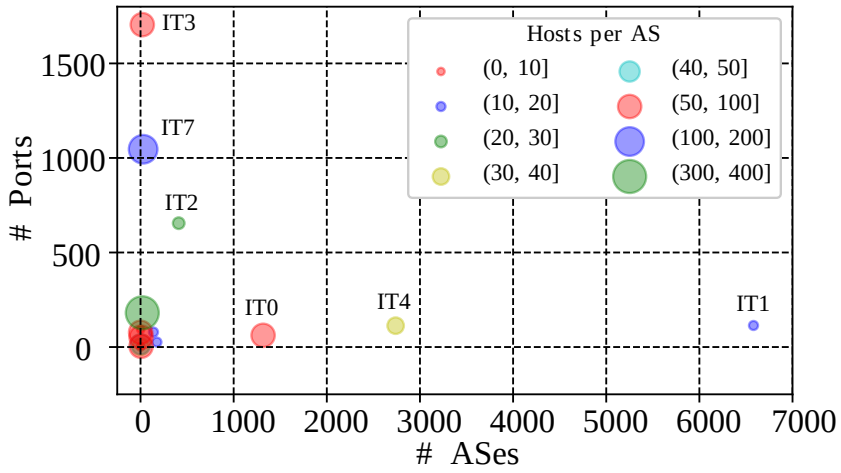
Focus now on communities IT3 and IT7 in the same figure. Those communities are formed by a low number of ASes. However, these ASes present a large number of IP addresses. More interestingly, the communities target a large number of destination ports. In other words, I record a small number of ASes in which a large number of IP addresses contribute to scan thousands of destination ports. These characteristics are typical of *horizontal scans*, such as those performed by tools like

Table 6.1: Basic statistics per community - IT

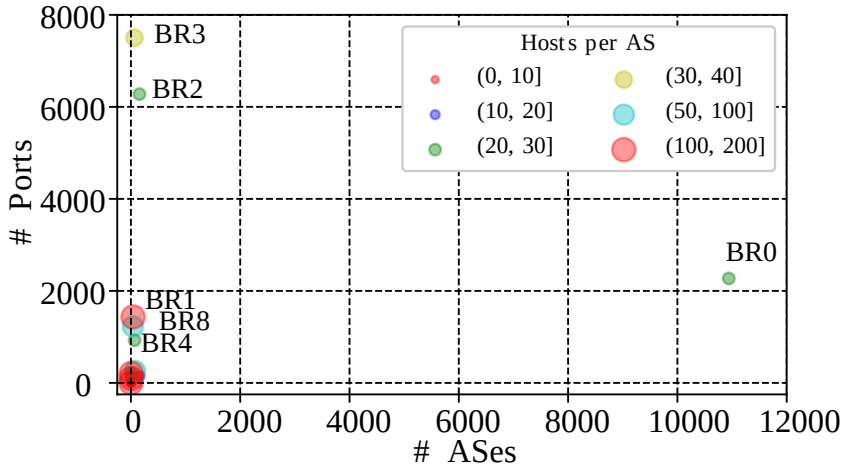
Community	# Ports	# ASes	# server IPs	# client IPs
IT4	895	2739	760	92122
IT0	222	1316	760	88864
IT2	1426	410	760	8449
IT3	2717	19	760	1072
IT1	439	6581	760	118269
IT7	1519	28	760	3073
IT8	401	19	760	6490
IT6	149	176	760	2921
IT9	105	6	760	359
IT5	89	139	760	2353
IT12	121	6	760	366
IT13	109	2	760	11
IT16	30	2	760	92
IT10	1	1	384	1
IT15	17	1	760	100
IT14	6	1	760	32
IT11	6	1	6	3
IT17	1	1	102	1

Table 6.2: Basic statistics per community - BR

Community	# Ports	# ASes	# server IPs	# client IPs
BR1	4283	156	8192	3123
BR0	2632	10940	8192	267880
BR2	13637	68	8192	2035
BR8	12513	40	8192	5841
BR3	18053	65	8192	2549
BR4	4628	36	8192	1873
BR6	417	170	8192	2424
BR9	796	75	8192	5240
BR10	450	7	8192	150
BR7	341	5	8192	21
BR15	454	6	8192	1051
BR12	1721	6	8192	633
BR11	488	9	8192	196
BR5	21	1	2638	176
BR16	4	2	8192	13
BR18	12	1	8192	4
BR19	4	1	6168	4
BR17	1	1	8190	1
BR14	2	1	1876	2
BR13	1	1	1	1



(a) Italy



(b) Brazil

Figure 6.5: Structure of communities in the Week 1.

nmap[4], likely run on servers hosted in specifc ASes.

The remaining communities match patterns where a small number of ASes targets specific ports. Those cases include port numbers usually not used by very popular services and ports not exploited in common attacks. This signature hints to sources performing sporadic scans, misconfigurations and other rare anomalies.

Interesting, the bottom plot in Figure 6.5 confirms results with data from the

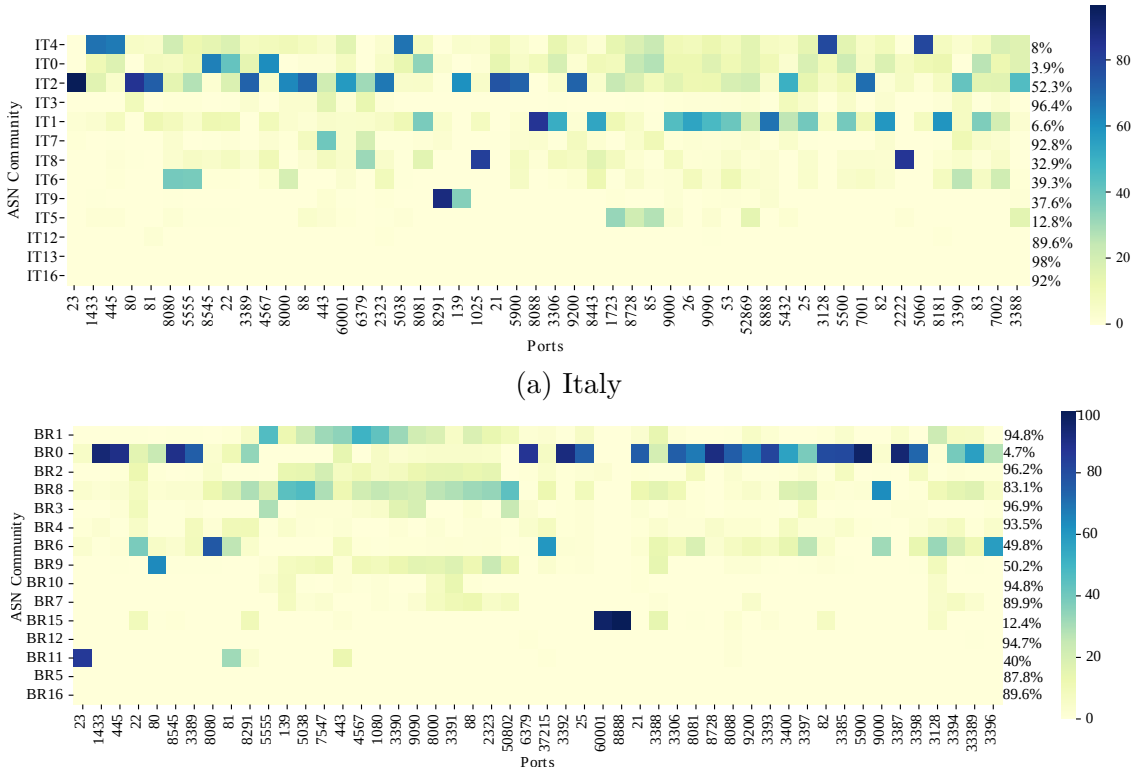
[4] https://nmap.org

Brazilian darknet. In this case, however, the number of communities in some sectors of the plot is reduced. Manual inspection shows that some of the communities seen in the Italian darknet may be merged by the modularity algorithm when applied to the (larger and noisier) Brazilian dataset. Other communities are completely diverse. Some source ASes are observed only in one of the datasets and target completely different sets of destination ports. These results confirm previous works [137, 122] that show dissimilarities in traffic reaching darknets deployed at diverse IP ranges.

6.4.3 Ports per community

Figure 6.6: Example output of the community detection algorithm.

I next dig into the ports each community targets as a way to understand the services these communities are probing. First, I visually explore the communities in search for patterns. A sample visualization is provided in Figure 6.6: It reports a sample of the communities for the Italian darknet. Node sizes are proportional to their degrees. The figure shows several behaviors. The community in lilac is an example of targeted action from distributed ASes. It gets represented as a strong concentration around two destination ports, namely ports 23 and 2323. Hundreds of ASes are connected to those nodes. These are ports usually hosting terminal services such as Telnet, drawing the attention of some sources by those services. A similar consideration holds for the community in green, for which ports 1 433 and



(a) Italy

(b) Brazil

Figure 6.7: Percentage of packets per Port to ASN communities.

445 are the most common targets. The community in blue, on the other hand, may indicated an horizontal action carried over by groups of ASes, likely scanning hundreds of ports.

The heatmaps in Figure 6.7 extend the analysis. They show with colors how the traffic for the top-50 ports ($x-axis$) is distributed according to the several communities ($y-axis$). The rightmost column of the plot quantifies, for each community, the percentage of traffic that is *not* related to the top-50 ports. Communities and ports are sorted according to their popularity.

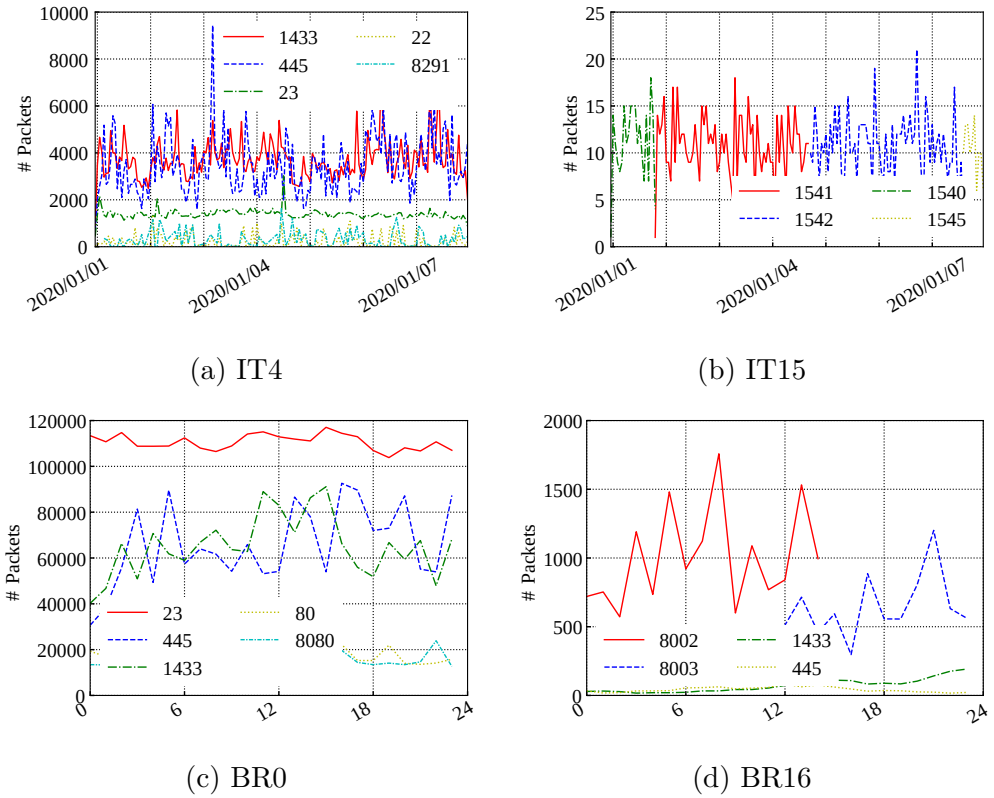
Focusing on the most popular community (IT4) at the top plot (Italy), observe how 92% is targeting several ports in the top-50. The community in particular contacts five ports that often host services vulnerable to remote attacks, such as MsSQL (port tcp/1433) and Microsoft Active Directory (port tcp/445). The second most popular community (IT0) is also strongly concentrated on the top-50 ports, differently from IT4, (e.g., port tcp/22), with only 3.9% of its traffic goes to the remaining ports.

The community IT2 is the clearest example of sources performing large-scale

horizontal scans for popular services. It dominates many of the top-50 ports, including services such as Telnet, HTTP, HTTPs, among others. Yet, notice how half of the traffic from this AS community is directed to ports not in the top-50 set. Finally, notice community IT9, which focuses mostly on port 8 291, related to a vulnerability on MikroTik RouterOS Winbox. This targeted behaviour is also visible in other communities that do not dominate any port in top-50 (e.g., IT3, IT5 and IT12).

The bottom plot (Brazil) confirms the general division of communities in (i) vertical scans for popular services – e.g., BR0; (ii) horizontal scans on multiple ports – e.g., BR2 and BR6; and (iii) targeted activity – e.g., BR11. Yet, here we clearly see major differences in formation of communities when compared to the Italian dataset. Notice, for example, how the port tcp/23 (telnet) is dominated by a single community (BR11), whereas it is dominated by sources performing horizontal scans in the Italian case.

6.4.4 Temporal behaviour



(a) IT4

(b) IT15

(c) BR0

(d) BR16

Figure 6.8: Time patterns of the top-5 most active ports in selected communities.

Finally, I investigate how sources in different communities behave over time. Figure 6.8 reports timeseries of the number of packets per hour for different ports in a community. I only show four of the most interesting communities and report activity for their five most active ports. Figure 6.8a and Figure 6.8b show results for IT4 e IT5 covering 1 week. Figure 6.8c and Figure 6.8d show results for BR0 and BR16 covering 1 day to improve visualization.

Recall from previous sections that IT4 is the most active community and vertically focuses on popular services. Figure 6.8a confirms, among the top-5 most targeted ports, the prevalence of ports 1 433 and 445, with a lower amount of traffic to ports 22, 23 and 8 291. I see that sources in this community produce a constant amount of noise. A similar level of traffic on each port reaches the darknet without any apparent daily or weekly patterns.

Figure 6.8b, on the other hand, shows a targeted community. Here the traffic volume is much less prominent, with only few dozens of packets reaching the darknet in each hour. The ports in the $[1\,540 - 1\,545]$ range are often used by a cluster management framework (RDS services). Notice how the few sources participating in this community have an orchestrated behaviour, alternating the contacted ports after some days of activity, which may suggest the presence of a coordinated, low-rate action. Similar results hold for the other two analyzed weeks, not shown for brevity.

Figures in the Brazilian darknet follows very close patterns. Figure 6.8c shows the hourly pattern for the 5 most contacted port in BR0 – an example of community engaged in vertical scans. Figure 6.8d shows instead BR16, an example of targeted community. Similar patterns as for the Italian case emerge in both cases. Notice however that sources in BR16 alternate ports with much higher frequency as before.

6.5 DPIPot validation

I further validate the methodology on the DPIPot setup, to check the compatibility of the algorithm with active deployments. Refer to Chapter 5 for a thorough characterization of the traffic reaching the active infrastructure.

I consider each of the top-5 protocols requested on AcmePot, namely RDP, HTTP, TLS, SSH and MsSQL-TDS (cfr. Tab. 5.5) separately. For each of the considered protocols, I observe a tail of ports receiving less than 10 flows during the analyzed month. Therefore, I choose to remove flows directed to such ports. Table 6.3 shows the result of this filtering process. Note that, despite discarding at least two thirds of the observed ports, the number of flows hitting the deployment still remains almost untouched (less than 10% reduction), and all the IP sources are still visible.

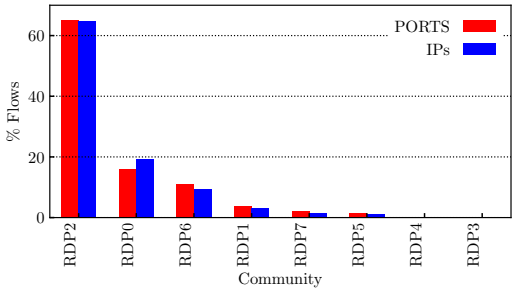
I build the graph as described in Sec. 6.2. Given the significantly lower number of sources requesting the L7 services, I can avoid to map each source IP address to

Table 6.3: Top-5 protocols recognized in DPIPot after port filtering. In brackets the percentage of retained ports and flows.

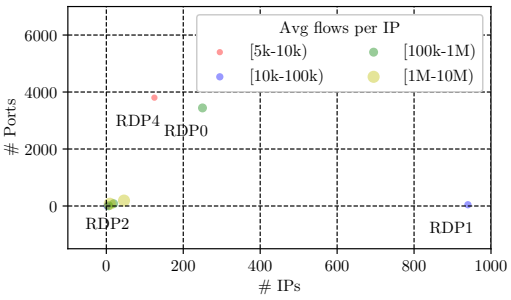
Protocol	Flows	Sender Addr.	Dest. Ports	# Comm.
RDP	329 549 604 (99.9%)	1 415	7 646 (26.9%)	8
HTTP	430 026 (96.4%)	13 705	3 191 (34%)	21
TLS	205 607 (92.8%)	2 806	2 876 (16%)	14
SSH	119 389 (99.7%)	1 097	30 (16%)	11
MsSQL-TDS	30 968 (97.1%)	3 193	7 (1.5%)	3

its AS and still maintain the graph manageable.

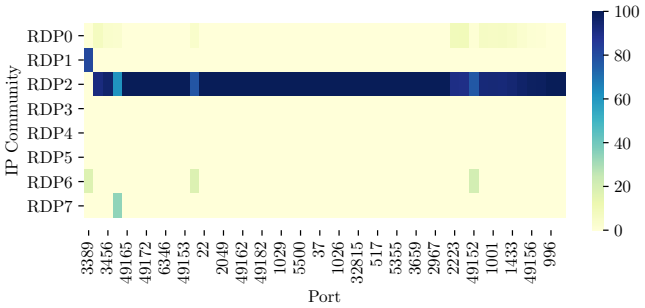
For the sake of space, I comment here only the results regarding RDP, the most common protocol in the trace. Figure 6.9 summarizes the flow distribution, community structure and flow destination. From Figure 6.9a I observe that RDP2 is the community gathering the largest number of flows. Figures 6.9b and 6.9c show that this community includes several hundreds of IPs, respectively producing high volumes of traffic (up to an average of 10 million flows per address) in an horizontal scan towards several hundreds of ports.



(a) Distribution of flows per community.



(b) Community structure.



(c) Percentage of packets per Port to IP communities.

Figure 6.9: RDP community breakdown.

RDP1 represents the opposite case: it gathers a low number of flows (less than 10%), sent by a large number of IP sources targeting a restricted set of ports for an average of 10 thousand times each. Figure 6.9c shows that IPs belonging to this community are mostly targeting the well-known port for RDP service - port 3 389 - in a vertical scan. In between cases are the smaller communities RDP0 and RDP4. The former gathers almost 20% of the total RDP flows, while the latter is significantly smaller ($< 1\%$). They include more than 3 000 ports each, but those ports are not shown among the top-50 ones.

6.6 Conclusion

In this chapter I presented a community detection-based methodology aimed at easing the analysis of large amounts of unsolicited traffic. Thanks to it, I was able to detect coordinated events, such as network scans due to botnets. The described methodology has automatically identified and isolated sources engaging in three major categories of events: vertical, horizontal and targeted scans.

This work is a study of the application of community detection algorithms to unsolicited traffic analysis. Many promising directions emerge. I plan to deepen the investigation on sources behaving similarly. In the same line, I plan to characterize the ASes hosting sources sending packets to darknets. Finally, I observe that some activities in darknet and honeypots are rather constant, thus becoming less relevant. I plan to apply advanced complex network approaches to filter out the *expected* noisy traffic, so to highlight events that are rare and potentially more interesting for cyber-security applications.

Chapter 7

Anomaly Detection Techniques on Network Traffic Time Series: Practical Considerations and Challenges

The methodology I present in this chapter is inspired by my paper *Regular Pattern and Anomaly Detection on Corporate Transaction Time Series*, presented at the 4th International workshop on Data Analytics solutions for Real-LIfe APplications (DARLI-AP 2020) [136]. A thorough review of classical and new AI-based anomaly detection techniques can be found on our book chapter *The New Abnormal: Network Anomalies in the AI Era* [138], published in *Communication Networks and Service Management in the Era of Artificial Intelligence and Machine Learning* [162].

7.1 Introduction

Authors of [34] define *anomaly detection* as the "problem of finding patterns in data that do not conform to expected behavior". In computer networks anomaly detection techniques have been employed in several tasks, such as finding nodes compromised by malware, triggering alerts in network monitoring systems and pinpointing faults reported in service logs.

Most anomaly detection algorithms used in networking problems are based on techniques proposed for other scenarios. Methods to perform anomaly detection are indeed researched and exploited since decades before the development of the Internet itself – from the study of outliers in probability distributions to the search for frauds in pre-Internet systems, e.g., banking systems. The surge on data coming

from networked applications (e.g., social networks, IoT devices, cyber-physical systems) together with the measurements needed to operate these applications have pushed anomaly detection further. Anomaly detection more than ever requires techniques and algorithms able to uncover anomalous behaviors on datasets that are large, complex and diverse.

Initial approaches ported to the network anomaly detection problem have been strongly rooted in rules-of-thumb, statistics, information theory and machine learning. Threshold-based anomaly detection, for instance, has been widely adopted in cyber-security for the detection of port scans and DDoS attacks. Similarly, diverse statistical solutions have been employed to identify anomalies on time-series exported by Internet telemetry systems. As more and more data became available, data-driven solutions gained momentum. Machine learning algorithms for prediction, clustering and classification have been applied on anomaly detection thanks to their good capabilities to automatically learn patterns from data. The trend has been exacerbated in recent years: The continuous growth on data availability, the unprecedented increase on computing resources and breakthroughs on AI research have allowed data-driven solutions to solve new complex problems on various fields. Some of these breakthroughs have potential to revolutionize the research on anomaly detection too.

This chapter focuses on the development of a sound anomaly detection benchmark in a network traffic scenario. Given different time-series reporting the traffic volume per time instant, I require the tested algorithms to detect anomalous patterns, i.e., unusual peaks, mean shifts, any deviation in the signal periodicity and other similar phenomena. I test several *classical* anomaly detection approaches against a more recent representation learning technique, the autoencoder. I want to investigate whether similar AI techniques add any advancement to the anomaly detection task. I discuss the upsides and downsides of each implementation, bearing in mind a possible practical application (e.g., a real-time network monitoring dashboard).

The reminder of this chapter is organized as follows: Section 7.2 includes some of the most relevant related works in the field and provides the lexicon that I will use throughout the chapter; Section 7.3 describes the basic functioning of each of the benchmarked algorithms; Section 7.4 provides details on the tested datasets, the full ML pipeline and the metrics I used to evaluate the unlabelled case; Section 7.5 describes the results, Section 7.6 concludes the chapter.

7.2 Problem characteristics and definitions

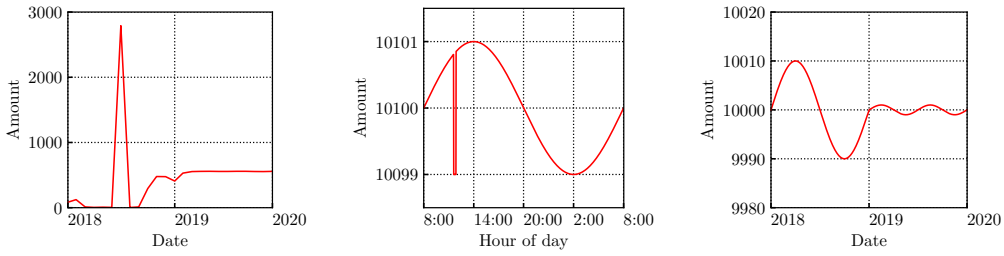
Anomaly detection in networked applications are studied since early days of the Internet. Many surveys have summarized the developments in the field [24, 95, 91, 6]. In this section I introduce the definitions and the taxonomy used throughout

the chapter, which is based on [34].

The term *Anomaly detection* aggregates many different, yet related, tasks: *Outliers* are values detached from the remaining samples. For example, given a random variable, an outlier can be a value that should not happen because it is out of the acceptable variable range, or because it falls far from the expected value. *Rare events* are usually defined similarly – points falling far from expected values. However, they represent events that are known to happen rarely. As such, some anomaly detection algorithms may consider such events as *normal*, even if they deviate from common patterns. Finally, *novelty* represents a behavioural (and possibly permanent) change of a variable. Unlike outlier detection, where deviating points have been seen before (in training), novelty detection aims at capturing whether a new sample is an outlier compared to the past or not. Here again the surge of a novelty can be considered an anomaly, as the novel points diverge from the usual patterns. Some novelty detection algorithms however try to identify whether points deviating from expected patterns represent indeed such a change in behaviour, thus tagging the change as novelty, rather than an anomaly.

In this chapter, I adhere to the somehow loose definition by authors of [34] and consider an *anomaly* any unexpected behavior in a data variable. Novelty detection, outlier detection and rare event detection are some of the related tasks that are commonly found in the literature, which I group together as anomaly detection.

Moreover, anomalies are classified as pointwise, collective or contextual [6, 24, 119], regardless of the nature of data. Examples are provided in Figure 7.1 considering a numeric attribute forming a time-series.



(a) Pointwise (b) Contextual (c) Collective

Figure 7.1: Examples of anomaly macro-categories.

Pointwise anomalies are individual data instances that diverge in a dataset. We see an example in Figure 7.1a, in which a single spike deviates from the regular behavior of the series. Examples of pointwise anomalies in network *security* and *performance* domains are (i) an abrupt rise in the number of packets reaching a server during a DDoS attack and (ii) increases in the RTT between two networks due to a temporary congestion.

Contextual anomalies are cases where an instance becomes anomalous thanks to its context, even if it would not be anomalous in isolation. Figure 7.1b provides an example. A single low value appears while the series is reporting (smooth) high values for the attribute. However, low values are expected to be seen in other contexts. In network *accounting*, such behavior could represent the bytes per hour in a backbone link during a short daytime outage. Yet, low values would still be expected during night periods, thus characterizing the contextual anomaly in the former case.

Collective anomalies are cases in which various data instances, in conjunction, form the anomaly. Figure 7.1c provides an example. Here a time-series that periodically oscillates suddenly changes trend. Whereas the newer values remain in the same range, they collectively change the series behavior. This is a classic example sometimes considered as *novelty*, as the changed behavior may become the new normal after the anomalous transition. A permanent decrease or increase in traffic volume in a link caused by a *fault* on peering links could produce a similar anomaly.

The datasets I choose in this chapter - as I will detail in the following - are composed by univariate time series, collected at different time granularities and including examples of pointwise and collective anomalies, with different levels of detection difficulty.

7.3 Tested algorithms

This section describes the algorithms I adopted for my benchmark. Table 7.1 summarizes the set of adopted approaches, their macro-categories and the required input dataset, namely simple timeseries, temporal features from past samples, or sample windows. I provide a brief explanation of every algorithm in the following.

Table 7.1: Benchmarked algorithms

Category	Algorithm	Acronym	Input dataset
Timeseries Forecasting	Auto Regressive Integrated Moving Average	ARIMA	Univariate TS
Supervised Regression	Support Vector Regressor	SVR	Temporal features
	Stochastic Gradient Descent Regressor	SGDR	Temporal features
	Decision Tree Regressor	DT	Temporal features
	ADABoost Regressor	ADA	Temporal features
	Random Forest Regressor	RF	Temporal features
Representation Learning	Autoencoders	AE	Windows

7.3.1 Classical approaches

Anomaly detection has been faced with multiple **machine learning** and **data mining** algorithms. In fact, most classic algorithms used for classification and clustering can be applied on anomaly detection too. As for classification algorithms,

for example, anomalies can be detected by training a model to recognize the normal or anomalous instances. Clearly, labels are needed for training these supervised algorithms, thus limiting their applicability. In some cases, to partially solve this issue, only normal instances are labeled, forcing any testing instance not assigned to a class to be marked as anomalous. In the case of clustering algorithms, data points are split into clusters based on arbitrary distance measures, which may be problem-specific. Points belonging to small clusters as well as those left unassigned are considered possible anomalies.

Statistical anomaly detection is instead based on the assumption that normal instances can be mapped to a stochastic model. Algorithms in this category mark as anomalies, instances that deviate partially or completely from the model. A classic technique belonging to the category is the so-called *boxplot rule*, which marks data instances as outliers considering a reference probability distribution.

Many algorithms have been derived from the **information theory** research. These techniques exploit different measures to quantify the information in a dataset, with the entropy being the most well-known alternative. For example, when considering entropy, some algorithms assume that normal instances would present attributes with a relatively low entropy, whereas the introduction of anomalies would cause an increase in entropy.

Several other categories of anomaly detection algorithms have been documented in the literature, and readers are invited to refer to [5, 7, 89, 6, 91, 24, 119] for a deeper discussion on them.

I choose to include in my benchmark a set of well-known [147, 120] algorithms for time-series forecasting and supervised regression techniques of easy usage and interpretation. In the following I describe the specifications about each implementation, its parameters and threshold criteria.

Auto Regressive Integrated Moving Average (ARIMA)

Time Series Forecasting is an extensively used technique to tackle the anomaly detection problem ([113], [60], [145], [81]). Its final objective is to provide a prediction of the future values of a time series based on its past values. By defining a reasonable confidence interval for the predicted values, I identify as anomalies the points that fall outside such predicted interval. Here I focus on ARIMA Models. Every ARIMA model requires as input a stationary time series, with three fundamental parameters: p, the number of autoregressive terms, d, the order of the differencing term and q, the number of moving average terms. As a general best practice, the values of p, d and q should usually kept below 3. I therefore run a grid search with parameters ranging from 0 to 3. At each iteration, I instantiate an ARIMA model per each combination and I chose the best model by calculating the Mean Squared Error between the actual value and the prediction yielded by every (p, d, q) triplet. Once I find the best model, I let it predict the following time

instants, then compute the upper and lower boundary of the confidence interval for each prediction, and I flag as anomalous every point falling outside such boundaries. In a nutshell, I consider anomalous every point consistently deviating from the ARIMA forecast result.

Supervised regression algorithms

Table 7.2: Lags dataset

Field	Value
y	Target variable
date	Date, minute granularity
y_t-D	Value of y at time bin b-B (B={1,..,6})
hourly_avg	Average value of y per hour
daily_avg	Average value of y per day
prev_H_hours	Value of y at hour h-H (H={1,..,3})
neigh_4_bins	Average value of y at -4 time bins

All the following techniques use as input the dataset described in Table 7.2, properly standardized. This dataset includes the target variable to predict - y, i.e., the volume of traffic in a given time instant - the reference timestamp, and a complete set of time-based features (e.g., the value of the target variable in the previous N time bins, the hourly/daily average value of the target variable, etc.). Similarly as before, I consider well-accepted ML algorithms trained to predict the next value $\hat{y}$. I run hyperparameter selection, and compare the prediction $\hat{y}$ with the actual value y. Again, if the predicted volume for a given time bin deviates consistently from the real one, the time bin is flagged as anomalous. Differently from ARIMA models, I do not have a standard way to compute confidence intervals, thus I rely on domain knowledge heuristics to flag outliers. In details I define two threshold-based control criteria to label a point as anomalous: the anomaly is advertised if at least one criterion is triggered. I list the criteria below:

- Criterion 1 (multiplicative):

$$\begin{cases} \text{Anomaly} : \text{if } (y > \tau\hat{y}) \wedge (\mid y - \hat{y} \mid >= \sigma) \\ \text{OK} \qquad\quad : \text{else} \end{cases}$$

- Criterion 2 (additive):

$$\begin{cases} \text{Anomaly} : \text{if } (y - \hat{y}) > k\sigma \\ \text{OK} \qquad\quad : \text{else} \end{cases}$$

Where τ and k are multiplicative thresholds I manually tune and σ is the standard deviation of the label over the last N training samples label.

For the regression, I consider state of the art algorithms. I perform a Grid Search with Cross-Validation for the parameter of each module, choosing the combination that minimizes the Mean Squared Error. For the Support Vector Regressor ([148], [53]) I evaluate two different kernel functions: the polynomial and the RBF one. All kernels require a regularization parameter C, evaluated between 10^{-7} and 10. The remaining parameters keep their default values. The Stochastic Gradient Descent Regressor [84] exploits the standard concept of stochastic gradient descent to fit linear regression models. I test the same values for the tolerance parameter C for this technique as well, combined together with two different loss functions (i.e., `squared loss` and `huber`) and four different learning rates (i.e., `constant`, `optimal`, `invscaling`, `adaptive`). I then exploit a set of Decision-Tree based regressors: I first optimize the parameters of a simple Decision Tree (*criterion*, i.e., the function to evaluate the quality of a split, *max_depth* and *min_samples_split*). I then use the resulting optimum as a building block for an AdaBoost regressor [126] and a Random Forest regressor [27]. For both I evaluate *n_estimators* spanning from 10 to 100.

7.3.2 Representation learning and autoencoders

The advent of deep learning and its automated feature learning abilities has opened the way to advances on representation learning. This latter includes the vast collection of techniques that directly or indirectly allow to learn rich features or representations from unstructured data [21]. Applied to anomaly detection, the idea would be to *constrain* the learned representations to produce a latent space where normal and anomalous (or novel) samples can be easily separated.

A number of recent work follows the representation learning ideas. Authors of [1] propose to augment the above reconstruction error approach with an additional *surprisal* metric, which assesses how likely a representation should occur under the learned model. The authors argue that detecting anomalies can leverage two approaches: (i) the ability to *remember* what has been seen and (ii) the ability to spot novelties, i.e., *surprisal*. They propose a novelty score that incorporates both. For the first, they leverage the reconstruction error. For the second, they learn an autoregressive model on the latent vectors of the autoencoder and use the resulting likelihood of the latent vector as a proxy for surprisal.

On the same line, authors of [88] leverage the learning of latent representations of normal instances in multiple domains, and the learned boundaries between normal and anomalous in some specific domains (for which they have a ground-truth) to *transfer* anomaly detectors from source domains (supervised, known) to target domains (unknown).

A somewhat similar intuition has been used for multi-view anomaly detection [78] where data instances can have multiple views – e.g., a video represented by audio, video and subtitles; a face that has multiple views; pages that have

versions in different languages. The intuition is to have multiple views of normal data instances generated from the same latent vector, while data instances that are anomalous shall have multiple latent vectors.

Another related approach has been proposed by the authors of [66] for anomaly detection in images. They train a classifier to distinguish between a set of geometric transformations applied to images. The learned representations in this auxiliary task is useful to detect anomalies at testing phase, by analyzing the output of the model when applied on transformed images.

Authors of [23] couple a similar approach with a *student-teacher* framework for unsupervised anomaly detection and pixel-precise anomaly segmentation in images. While the teacher network learns latent features from a set of images, an ensemble of student networks is trained to regress the teacher's output on anomaly-free input. When fed with data with anomalous parts during the testing phase, the student networks will exhibit higher regression errors and lower predictive certainties in areas involving anomalies.

A famous representation learning technique is Autoencoders. These are neural networks that compress the input data into a latent space and, then, reconstruct the input based on the latent variables. Figure 7.2 depicts the basic idea: considering X as input, the encoder (left) compresses the input to a latent space $h = f(X)$. The decoder (right) reconstructs the input based on h, with $X' = g(h)$. The quality of the reconstruction is then evaluated by means of the *reconstruction error*.

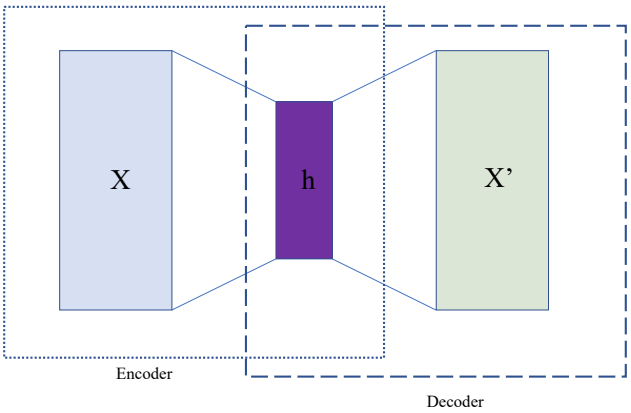


Figure 7.2: Basic autoencoder structure.

The autoencoder is trained with normal instances for anomaly detection. Then, the reconstruction error grows when the autoencoder is fed with anomalous instances during the testing phase, pointing to anomalies. Autoencoders are largely

used to detect anomalies in images, videos and text, but have found applications in several other scenarios too. For example, authors of [161] propose a deep autoencoder – called Robust Deep Autoencoder – that not only discovers high-quality, non-linear features from input instances, but also eliminates outliers and noise without access to clean training data. The model takes inspiration form Robust Principal Component Analysis, as defined in [32]. It aims at splitting the input instances into two parts: a low-dimensional representation of the input data, that can be effectively reconstructed by a deep autoencoder, and another one that contains element-wise outliers.

Authors of [36] highlight challenges for the application of autoencoders on anomaly detection, in particular the sensitiveness of the technique to noise and the need for large training sets. The authors then propose RandNet, an ensemble of autoencoders relying on different NN architectures for outlier detection. In a somehow similar direction, authors of [163] present the Deep Autoencoding Gaussian Mixture Model, which combines a compression network with an estimation network. The joint optimization of the two networks is claimed to improve performance of unsupervised anomaly detection on multivariate, high-dimensional data.

The assumption that autoencoders produce large reconstruction errors for every anomaly is questioned in [67] and others [163, 160, 72]. Some anomalies may be subtle, and the autoencoders may generalize normal instances to the point of overlooking anomalies. Authors propose the Memory-Augmented Autoencoder, i.e., MemAE, which memorizes prototypes of normal instances. In testing phase, the network will always use one of the prototypes in memory for reconstruction, hopefully increasing the reconstruction error in case of anomalies.

Finally, a similar problem is targeted in [20], emerging when the dataset used for training the autoencoders is contaminated with anomalies (e.g., noise). The autoencoders could learn how to reconstruct the anomalous instances, reducing the reconstruction error for other anomalies. Anyway, as the dataset is by definition mostly composed by normal samples, observing the reconstruction error distribution helps in counteracting this effect.

The latter use case is also the one I target in this chapter, as also the training samples in the selected timeseries may contain anomalous data. For detecting the anomalous points, I follow the simple principle described above, i.e., I mark as anomalous the input samples whose reconstruction error deviates significantly from the error distribution.

Figure 7.3 depicts an example of how the original timeseries needs to be reshaped to serve as input for the autoencoder. Each time bin t_i becomes a part of an input window w_i containing a total of W time bins. Note that, as the window is sliding of one bin at a time, most bins are included in more than one input sample. Such windows are then stacked to form the final input matrix M. The window size W, together with the training *batch_size* and *epochs* is a paramount hyperparameter for the effectiveness of this technique. To optimize their choice I run a grid search

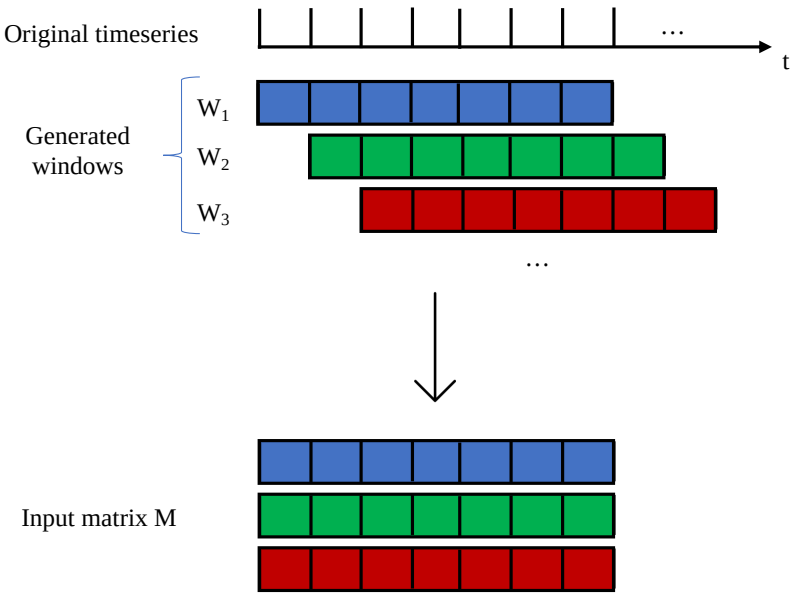


Figure 7.3: Input samples for autoencoders.

and I choose the triplet minimizing the validation loss.

Once the autoencoder is trained and validated, I test it against new input samples. Every time a new sample enters the autoencoder, it generates its reconstructed sample $\hat{w}_i$ and evaluates the mean squared error between w_i and $\hat{w}_i$. If the sample error overcomes the 95^{th} percentile of the error distribution, I mark as anomalous every t_i included in w_i.

7.4 Datasets and methodology

As previously mentioned, evaluating the effectiveness of anomaly detection algorithms is particularly difficult in case of unlabelled datasets. Before running the benchmark on DPIPot RDP data, I validate the process by means of an artificially generated dataset. Table 7.3 summarizes the composition of the tested datasets, ordered by increasing level of anomaly detection difficulty. For the Easy and Medium use cases, both provided with labels for anomalous points, I combine several instances of the well-known UGR'16 dataset [99]. This dataset aggregates a mixture of background traffic flows and artificially generated attacks. Each flow is summarized in the standard netflow format and provided with a label. Among the attacks, UGR'16 numbers various types of DoS attacks, port scans, botnet and

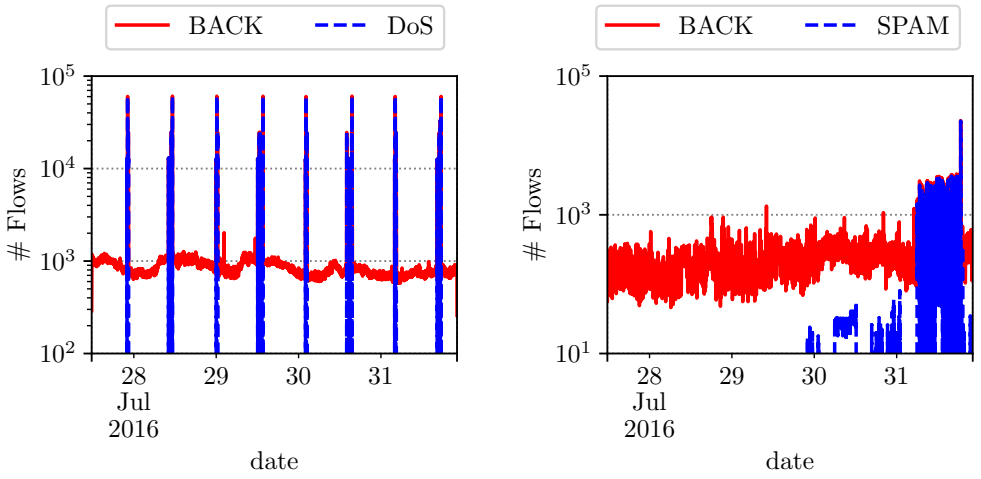
spam traffic. For my use cases, I chose to retain only the background traffic reduced of a factor of 10, firstly adding to it the One-to-One DoS attack traffic (cfr. Figure 7.4a, DoS spikes are evident and regular), subsequently adding single spam attack (cfr. Figure 7.4b, with a continuous spam attack at the very end).

Finally, Figure 7.5 depicts the RDP traffic profile captured on DPIPot (see Section 5.6.3 for a full characterization). Also in this case, I chose to artificially modify the total signal by adding and removing at random time instants the traffic produced by the first 10 heavy-hitters, as shown in Figure 7.5a. In total I perturb the original signal in 60 time bins (less than 1% of the total dataset). These time bins will be my reference in the absence of a proper ground truth.

Each of the chosen algorithms should provide its own prediction of the traffic patterns, and flag as anomalous the time bins deviating from it, ideally the ones in which the attack is present.

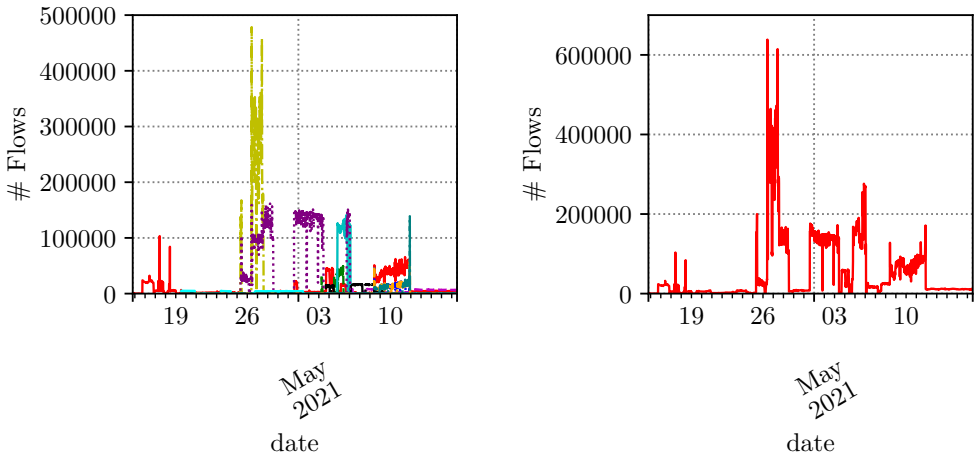
Table 7.3: Tested datasets

Name	Composition	Sample rate	Label	Type of anomaly	Difficulty	Dataset size
UGR'16	background+DoS11	1'	✓	Pointwise	Easy	6 388 points
UGR'16	background+Spam	1'	✓	Continuous	Medium	6 388 points
DPIPot	RDP	10'	×	Various	Hard	4 570 points



(a) Easy (b) Medium

Figure 7.4: Artificial datasets (y-axis in log scale) - anomalous samples are highlighted in blue.



(a) Hard case - Composition. (b) Hard case - Final signal.

Figure 7.5: RDP timeseries composition and final signal. Different colours indicate the activity of different heavy-hitter sources.

Every algorithm is trained and tested iteratively in a walk forward fashion, as depicted in Figure 7.6. At every iteration I_{i+1} the older samples (in red in the Figure) are dropped, and the newer samples (green) are included in the training even if they are marked anomalous in the previous iteration I_i. The idea is to mimic the incoming of new events in an online environment, where the framework should learn from the recent past and report significant changepoints to the network analyst.

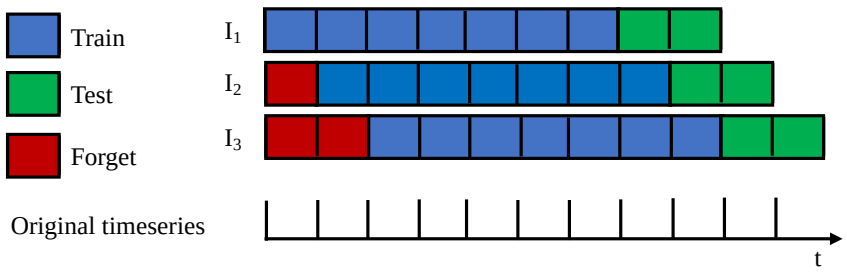


Figure 7.6: Walk forward process.

Finally, I need to define a set of metrics to evaluate the detection performance of each algorithm. For this task as well, I distinguish between the labelled cases (i.e.,

Easy and Medium) and the unlabelled one (Hard). In the former case, Accuracy and weighted F1-Score are enough to evaluate the detection capabilities. The latter case requires the definition of three new metrics:

- *A_tot*: total number of alarms, ideally the lower the better;
- *Correctness*: how many of the artificially introduced changepoints does the algorithm detect?
- *Consistency*: given the anomalies spotted on the timeseries without perturbations, how many of those are still present when we artificially modify it?
- *Mean Squared Error*: computed between the real and the predicted signal to evaluate the predictive capabilities of each algorithm.

Together with these quantitative metrics I take into consideration the visual distribution of the anomaly flags, as this aspect has a significant impact on a potential practical implementation of the whole pipeline. I discuss the outcome of each algorithm in the Results section.

7.5 Results

7.5.1 UGR'16 - Artificial dataset validation

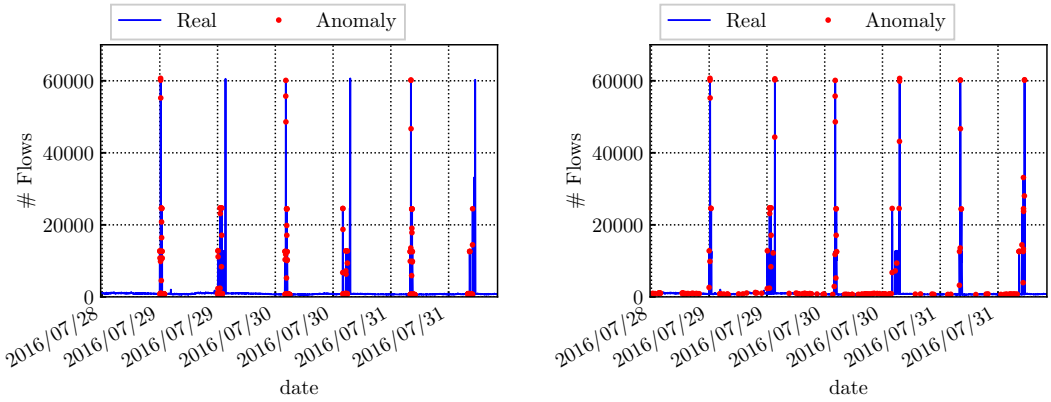
Table 7.4: Easy case - Detection performance

Algorithm	Flagged Points (%)	Accuracy	F1-Score
ARIMA	276 (5.58%)	0.944	0.954
SVR	388 (7.84%)	0.938	0.953
SGDR	389 (7.86%)	0.948	0.961
DT	272 (5.49%)	0.956	0.964
ADA	276 (5.57%)	0.951	0.96
RF	**205 (4.14%)**	**0.963**	**0.967**
AE	420 (8.84%)	0.937	0.953

I here show the performance of each algorithm in the UGR'16 use cases. Table 7.4 summarizes the classification metrics for the easy case. All the algorithms show satisfactory values for both accuracy (0.937 - 0.963) and F1-score (0.953-0.967). Random Forest (in bold in the table) is the one yielding the most satisfactory performance in term of total number of flagged points, accuracy and F1-score. Autoencoders, Support Vector and Stochastic Gradient Descent are the ones flagging the highest number of anomalies.

See Figure 7.7 for an example. Taking a look at the metrics, the autoencoder seems to be penalized by the required input format: the algorithm marks as anomalous all the time bins concentrated around the spikes, because the spikes are included in more than one input window (cfr. Figure 7.7a - note that many red dots are almost overlapping). This aspect does affect the accuracy and F1-Score, but the result can still be useful for a first qualitative interpretation of the signal: a network analyst could profit from a visualization of the autoencoder output, as it draws attention on restricted time intervals that are worth observing. Opposite considerations are valid in the case of RF (cfr. Figure 7.7b): despite the good metrics, visualizing a similar output is not helpful to isolate relevant events. Finally, both SVR and SGDR flag points across the whole signal without a specific rationale, leading to unreliable results.

All in all, in such a trivial scenario the adoption of easily tunable and interpretable algorithms - as the tree-based ones and Random Forest in particular - is enough to have a clear and reliable picture of the anomalous instances.



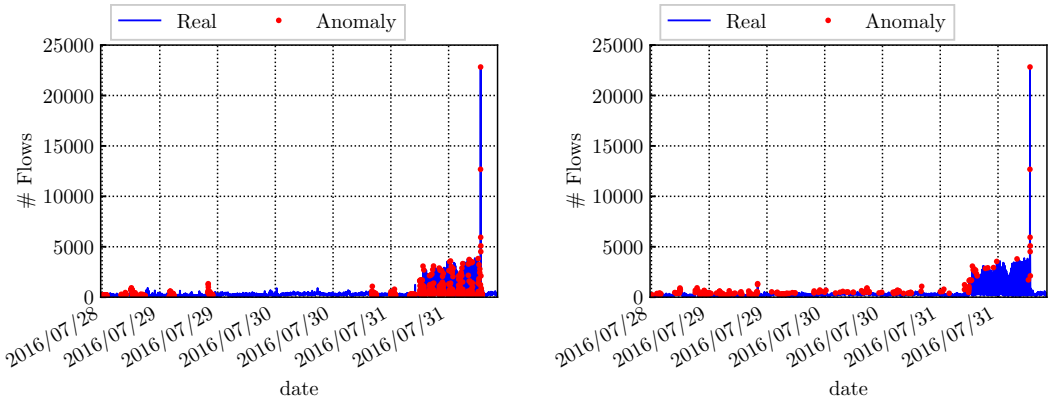
(a) AE (b) RF

Figure 7.7: UGR'16 Easy case - anomalous points flagged by different algorithms.

Table 7.5: Medium case - Detection performance

Algorithm	Flagged Points (%)	Accuracy	F1-Score
ARIMA	1 532 (30.96%)	0.62	0.641
SVR	384 (7.76%)	0.74	0.689
SGDR	365 (7.37%)	0.747	0.695
DT	288 (5.82%)	0.75	0.688
ADA	231 (4.67%)	0.757	0.69
RF	**198 (4%)**	**0.761**	0.69
AE	960 (19.4%)	0.726	**0.721**

Similar considerations hold for the medium case. As visible in Table 7.5, a more complex case yields generally lower values of accuracy (0.62 - 0.761) and F1-Score (0.641 - 0.721). Tree-based algorithms seem again to be the best trade-off between complexity and performance, with Random Forest outperforming the other algorithms in terms of accuracy and number of flagged points. Despite the large number of points flagged, again due to the input format, the autoencoder reaches the highest value of F1-Score. Also in this case the autoencoder output may aid the event interpretation in practice: comparing Figures 7.8a and 7.8b I observe how the Autoencoder concentrates all the flags around the anomalous area, while Random Forest flags are more scattered over time.



(a) AE (b) RF

Figure 7.8: UGR'16 Medium case - anomalous points flagged by different algorithms.

7.5.2 DPIPot RDP - Hard case

Table 7.6: Hard case - Detection performance

Algorithm	A_tot (%)	Correctness (%)	Consistency (%)	MSE
ARIMA	386 (12.45%)	14 (23.73%)	20 (13.61%)	$3.94 \cdot 10^9$
SVR	1247 (40.22%)	17 (28.81%)	645 (46.24%)	$1.16 \cdot 10^{10}$
SGDR	3026 (97.61%)	45 (76.27%)	1902 (99.27%)	$1.41 \cdot 10^{10}$
DT	505 (16.29%)	11 (18.64%)	**152 (31.73%)**	$2.54 \cdot 10^9$
ADA	399 (12.87%)	8 (13.56%)	139 (42.9%)	$2.02 \cdot 10^9$
RF	**348 (11.64%)**	9 (15.25%)	97 (27.95%)	$1.03 \cdot 10^9$
AE	407 (13.12%)	**16 (27.12%)**	74 (9.15%)	$1.75 \cdot 10^9$

Finally, I present the results in the Hard case. As mentioned earlier, the algorithm evaluation is not trivial in an unlabeled scenario. I look for the best trade-off among the metrics (cfr. Section 7.4) and the predictive capabilities of each algorithm, here evaluated by means of the MSE. Table 7.6 summarizes the results. By taking a look only at *A_tot* (i.e., the total number of alarms raised by each algorithm), I can already rule SGDR and SVR out of the reliable algorithms, given the high number of flagged points (higher than 97% in the case of SGDR). The poor performance of both algorithms is confirmed by the MSE, one order of magnitude larger than the other algorithms (i.e., higher than 10^{10}). The remaining algorithms show comparable values of *A_tot* (from 11.64% for RF - in bold - to 13.12% of AE). In order to rule out the best one I should refer to the remaining metrics: AE is the most correct one, while DT the most consistent. AE and RF are the ones yielding the lowest values of MSE.

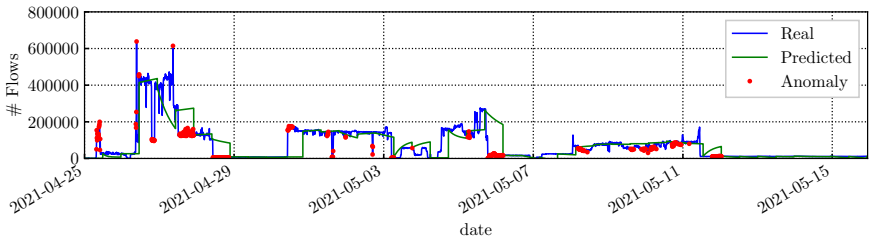
All in all, determining which algorithm performs best in such a complex case is not trivial and is strongly dependent on the practical needs of the final user. The visualization of the predicted signals compared to the real one may help in the choice. Figure 7.9 shows the four more relevant cases out of the seven selected algorithms. Figure 7.9a shows how ARIMA only predicts flat segments, following the mean shifts of the signal over time. This behavior can still be useful, if the purpose of the network analyst is only to be warned in case of significant changes in volume, but it leads to the neglect of smaller and potentially interesting events. Figure 7.9b, instead, confirms the poor performance of SVR, which proved to be unsuitable for this purpose.

RF and AE (Figures 7.9c and 7.9d) show good predictive capabilities and generate a relatively low number of alarms. Again, the flag distribution depends on the different format of the input: the AE flags are more concentrated around the sudden spikes, while the RF ones are more distributed and cover also flatter parts of the signal presenting minor oscillations. Both can be considered a good fit for the problem, but the best trade-off should be chosen based on the monitoring needs of the final user, or combined to have a full and clearer view.

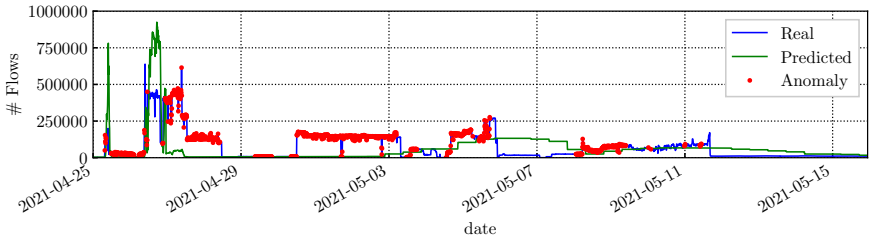
7.6 Conclusion

In this chapter I tested a set of well-known anomaly detection algorithms benchmarking their effectiveness on anomalous network traffic time series. I focused on highlighting the advantages and limitations of a new AI-based technique - the autoencoder - compared to more well-known techniques.

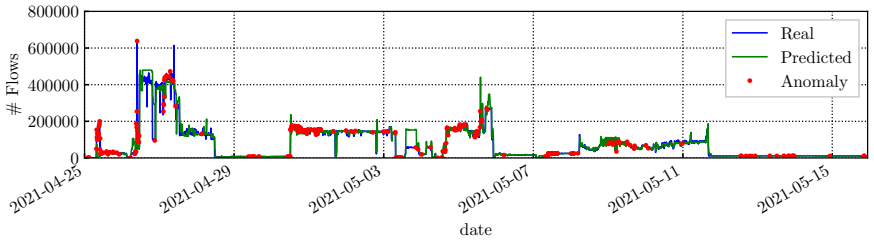
I first validated the whole pipeline by means of a well-known labelled dataset, defining two cases of increasing detection difficulty, and finally tested the methodology against real RDP flows reaching the DPIPot infrastructure. I artificially constructed a set of anomalies to be used as a reference, and defined three metrics



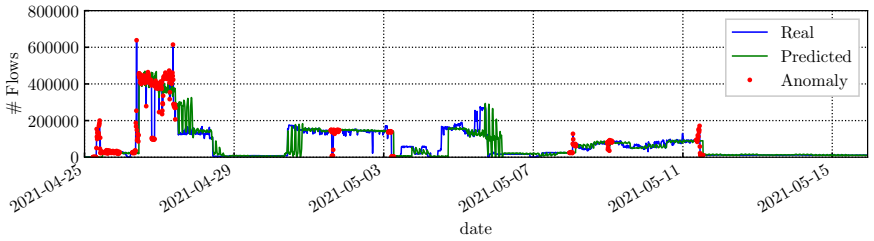
(a) ARIMA



(b) SVR



(c) RF



(d) AE

Figure 7.9: RDP hard case - predictions and anomalous points flagged by different algorithms.

to allow a fair benchmark. I showed that, in presence of easily recognizable anomalies (i.e., clear pointwise anomalies), all the tested algorithms give a satisfactory

performance, and therefore the most easily implementable and interpretable algorithm should be chosen. On more difficult cases (e.g., continuous anomalies) the performance of all algorithms decreases, and too simple ones should be considered particularly unreliable. Algorithms as the Random Forest and the Autoencoder guarantee the best values of accuracy and F1-Score respectively, and the choice of one rather than the other should be made taking into account the practical needs of the final user (e.g., minimize alarms, precision, easy visualization, etc.).

Similar considerations hold for the hard case as well: a one-size-fits-all solution does not exist, and the choice is strongly dependent on the monitoring requirements. Random Forest and Autoencoder are the best performing algorithms, but they flag points based on different principles. If one's interest is to detect and visualize sudden spikes and their immediate surroundings, then autoencoders should be preferred; otherwise, if small oscillations and continuous anomalies are the focus, Random Forest is the right choice. Combining the two outputs may be a viable solution, provided that the user agrees to tolerate a higher number of alarms.

Eventually, an aspect that is worth underlying is the need of a correct threshold definition. This is crucial for the correct functioning of the system, though it still requires an extensive domain knowledge to be properly set, especially when using classic approaches. Using an autoencoder and evaluating the error distribution to set a threshold allows a first step towards the full automation of this task, but still does not allow to completely forego the human intervention.

Chapter 8

Conclusion and Outlook

In this thesis, I presented several works addressing the analysis of unwanted network traffic. I used big-data and machine learning techniques to detect different patterns, communities and anomalous behaviors, providing a full pipeline useful for network analysts and cybersecurity practitioners, as well as for researchers. The key idea is to provide a complete toolkit automating the network monitoring task without the need of complex tuning or human intervention.

In the first part of my thesis I characterized relevant events taking place in different deployments, either completely passive or active at different levels. I highlighted some of the most requested services, evaluating their patterns and similarity in time and space. I demonstrated how a large part of the packets composing the Internet Background Radiation target darknets regardlessly their geolocation. I enriched the fully passive framework adding active deployments, registering the changes in patterns and requests when active targets are deployed in the network. I showed how responding at different layers, horizontally on all ports or vertically on selected services, attracts different types of requests and, when the senders have malicious purposes, leads to different attack phases.

In the second part, I described the results of the automatic aggregation of traffic sources, showing how aggregating the sources in form of a graph allows the execution of community detection algorithms that group together seemingly uncorrelated activities. At last, I discussed the effectiveness of several well-known and more advanced AI-based anomaly detection techniques. Bearing in mind the practical implications of my work, I mostly chose to focus on off-the-shelf methodologies, discussing their advantages and limitations in terms of scalability, interpretability and visualization.

All in all, I believe that the key findings of this work represent a step forward in the analysis of the internet background radiation. Firstly, the light shed on the way most services and ports are requested by remote sources may drive the practitioners towards a more effective configuration of the existing cybersecurity systems and to the recognition of otherwise unknown phenomena and actors. Moreover, the

proposed machine learning methodologies will surely speed up the analysts' work, and can be adapted to other monitoring scenarios.

Several research directions emerge from the encountered topics. The first one is the characterization of the remote sources observing their activity at the application layer: as I showed earlier, most of the active deployments receive some application payload, that often contains login attempts, malicious binaries, or command sequences. Having a deeper knowledge of such phenomena represents a further step towards the design of efficient cybersecurity systems.

Moreover, both the community detection and the anomaly detection task can benefit from newly deployed AI techniques. Apart from the autoencoders, recent works propose the application of Natural Language Processing techniques to the community detection task [64], as well as various Generative Adversarial Networks [127], Reinforcement Learning [77] and Deep Learning [159] techniques to detect anomalies in network traffic. Such methodologies allow the usage of more complex, multivariate datasets containing more complex features as well.

Appendix A

DPI Solutions in Practice: Benchmark and Comparison

The work I present in this chapter is mostly taken from my paper *DPI Solutions in Practice: Benchmark and Comparison*, presented at the 6$^{\text{th}}$ International Workshop on Traffic Measurements for Cybersecurity (WTMC 2021) [121].

A.1 Introduction

The internet is a continuously growing ecosystem composed by diverse protocols and applications. The rise and spread of smart devices, video-conference platforms as well as the continuous appearance of sophisticate cyber-attacks keeps changing the characteristics of traffic observed in the network. Understanding protocols that are carrying specific flows in the middle of such a variety of traffic has always been essential for multiple applications, in particular for those supporting network security like firewalls and IDS.

Deep Packet Inspection (DPI in short) has been the dominant approach to perform protocol recognition, showing effectiveness in several traffic monitoring scenarios. DPI parses traffic payload searching for signatures that characterize the protocols. Indeed, many DPI solutions do exist and still find important applications, despite the increasing usage of encrypted protocols. DPI is particularly useful in cyber-security scenarios, such as for intrusion detection systems, firewalls and other tools supporting security (e.g., flexible honeypots). The timely identification of a broad range of protocols remains a key first step in the security use case, calling for accurate, efficient and up-to-date DPI solutions. Yet, previous efforts providing an independent evaluation of DPI are already aged [30] or leverage on restrict traffic traces, which questions the applicability of such results to practical scenarios.

I revisit the question on the quality of DPI-based protocol identification. I select and evaluate four popular, open source projects implementing DPI, namely

nDPI [49], Libprotoident [10], Tstat [144] and Zeek [116]. I first study their classification using passively captured traces, covering a wide range of scenarios, i.e., traffic produced by IoT devices, collaborative platforms/video-calls, malware, as well as production internet traffic. Establishing a ground-truth is challenging when dealing with such diverse traces composed by dozens of protocols. I here evaluate the *consistency* of the classification provided by the tools, relying on heuristics and domain knowledge to validate the decision of each tool when finding conflicting cases.

After that, I investigate whether the DPI solutions operate consistently when exposed to a limited number of packets per flow. Indeed, network applications usually perform protocol identification on-the-fly using the initial packets of each flow, in order to take timely decisions. For this, I investigate the number of packets per flow each solution needs to reach a decision, as well as the consistency of such decisions as more traffic is observed.

The obtained results show that:

- All tested solutions perform well when facing traces with well-established protocols. This is particularly true for popular protocols that account for the majority of production traffic;
- Some DPI solutions struggle when facing unusual events, such as massive scans or malware traffic;
- All tested tools reach a final decision already after observing the first packets with payload in a flow;
- nDPI outputs labels more often than others, and it usually agrees with the majority when tools diverge about the protocol of a flow.

To foster further research and contribute to the community, the code and the instructions to build the complete datasets and repeat the experiments is shared.[1]

Next, Sect. A.2 summarizes the related work. Sect. A.3 introduces our datasets and methodology. Sect. A.4 describes the results, and finally Sect. A.5 concludes the chapter.

A.2 Related Work

DPI has been applied to protocol identification since the early 2000s, when the usage of well-known ports for traffic identification turned out to be unreliable. Multiple approaches have been proposed. Some works rely on "shallow" packet

[1] https://smartdata.polito.it/dpi-in-practice/

inspection [35], i.e., they parse only packet headers in the search for protocol fingerprints. Such techniques still find practical applications, as encryption protects protocol payloads. Others propose efficient approaches for DPI, e.g., using pattern matching [19] or finely-tailored DPI algorithms [90]. Finally, some works rely on stateful information from multiple flows to label traffic, e.g., leveraging the DNS to obtain the labels used to classify encrypted traffic [144].

Many DPI tools have been introduced implementing such techniques. Here I consider four alternatives, which have been evaluated by original authors in [10, 49, 116, 100]. In contrast to them, I perform an independent evaluation of the tools, thus providing also a validation of the authors' results.

Past works compare DPI solutions. Authors of [104] perform an extensive benchmark covering port-based classification, packet signature algorithms etc. In [156], authors survey approaches to overcome the lack of ground truth in such studies. In some cases manual labelling of packet captures is used for DPI comparisons [9], while other works rely on active measurements to enrich captures with information about underneath applications [140, 69, 98].

Closer to this analysis is the work presented in [30], where authors also provide an independent comparison of DPI solutions. In contrast to [30], I leave out of our evaluation proprietary tools and libraries, since the lack of source code makes it hard to explore and explain discrepant results. I also refrain from evaluating tools no longer maintained. More important, I provide an updated comparison of DPI tools considering recent and real traces, thus covering scenarios not evaluated in the previous work, with a particular focus on modern security applications.

A.3 Datasets and Methodology

Fig. A.1 summarizes our methodology. I describe the DPI tools selected for testing (Sect. A.3.1). Then, I build up a set of traces covering different traffic scenarios (Sect. A.3.2). Next I process the traces with the DPI tools. I then perform several steps and build up heuristics to build a reference label, and find discrepancies on the final classifications (Sect. A.3.3).

A.3.1 Selection of DPI Tools

I here restrict the analysis to DPI tools that perform *protocol* identification (e.g., HTTP, TLS, SSH etc.), ignoring those aiming at the identification of the services generating traffic (e.g., Google, Facebook etc.) [85, 3]. Namely, I focus on the following four alternatives:

- *nDPI* [49] is an open-source DPI library written in *C* and based on dissectors, i.e., functions that detect the given protocols. It is an *OpenDPI* [83] fork optimized for performance and supports more than 100 protocols.

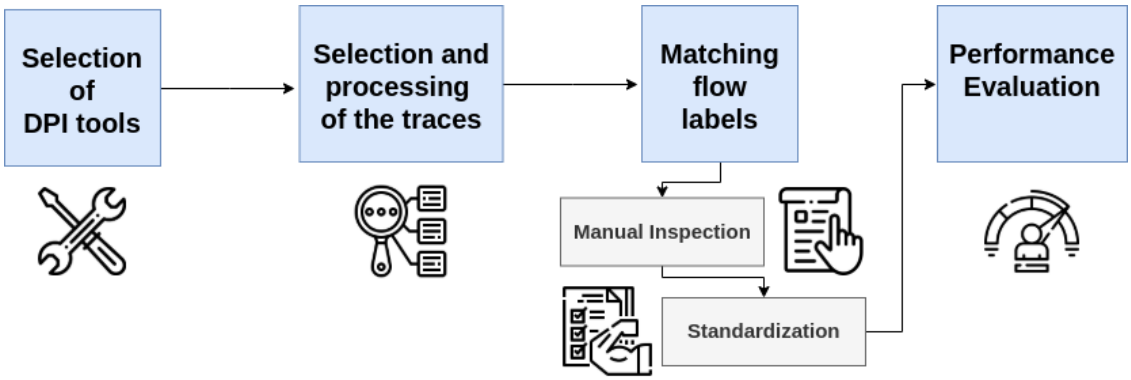


Figure A.1: Testing methodology.

- *Libprotoident* [10] is a *C++* library that focuses on L7 protocols. It applies a lightweight approach that uses just the first 4 bytes of payload. The idea is to overcome drawbacks of DPI, i.e., computational complexity and privacy risks. The library combines pattern matching with algorithms based on payload sizes, port numbers and IP matching. It supports over 200 protocols.
- *Zeek*[2] – formerly *Bro* [116] – is a complete framework for traffic analysis that also allows L7 protocol recognition. It exploits a combination of protocol fingerprint matching and *protocol analyzers*. It currently supports more than 70 protocols.
- *Tstat* [144] is a passive traffic monitoring tool that classifies traffic flows. It identifies a set of L7 protocols using payload fingerprint matching. It supports over 40 protocols.

Recall that I ignore projects no longer active. In particular, I leave *L7-filter* out since it has been shown to produce unreliable results in more recent scenarios [9]. Equally, I ignore proprietary alternatives, given the intrinsic difficulty to evaluate the root-causes of conflicting results without access to source codes [49]. Finally, I do not evaluate *tshark*[3] as it has proved much slower than the alternatives.

A.3.2 Selection and pre-processing of traces

I consider four scenarios to compare the DPI alternatives, including not only common internet protocols, but also protocols encountered by security applications.

I select 421 different PCAP traces that are aggregated in four macro-categories: (i) *User*, which includes ordinary browsing activity of ISP users while at home;

[2] https://zeek.org

[3] https://www.wireshark.org/docs/man-pages/tshark.html

Table A.1: Flows exported by the different tools before the pre-processing.

Macrotrace	Tool	Flows	
		TCP	UDP
User Traffic	Tstat	681 k	1.1 M
	Libprotoident	678 k	1.1 M
	nDPI	543 k	1.1 M
	Zeek	804 k	1.2 M
Media & Games	Tstat	15 k	16 k
	Libprotoident	15 k	14 k
	nDPI	10 k	21 k
	Zeek	17 k	16 k
Malware	Tstat	858 k	979 k
	Libprotoident	858 k	993 k
	nDPI	891 k	1 M
	Zeek	1242 k	971 k
IoT	Tstat	118 k	50 k
	Libprotoident	118 k	51 k
	nDPI	120 k	62 k
	Zeek	119 k	52 k

(ii) *Media & Games* [111, 45, 50] that includes conference-calls, RTC applications, multimedia and gaming traffic; (iii) *Malware* [130], which aggregates several samples of malware[4] and security experiments;[5] and *IoT* [132, 115], captured in different labs hosting a variety of IoT devices. I include both traces captured in our premises and third-party traces available on public repositories. Traces cover multiple years, and total more than 143 GB of PCAP files. For brevity, I do not provide details of each PCAP file here, instead describing only the aggregated *macrotraces*. To allow others reproduce these results, I link the public PCAP files in our website.[6]

I need to match flows as defined by each DPI tool for comparing their performance.[7] However, tools employ different rules for defining and exporting *flow records*. For example, each tool uses various timeouts to terminate flows that become inactive. Equally, traffic flags (e.g., TCP `FIN` and `RST` flags) are possibly used to identify the end of flows, releasing memory in the traffic monitor. The way such rules are implemented differs and, as a consequence, tools identify and report

[4] https://www.malware-traffic-analysis.net

[5] https://www.netresec.com/?page=PcapFiles

[6] https://smartdata.polito.it/dpi-in-practice/

[7] I use the classic 5-tuple definition for a flow: Source IP address, destination IP address, source port, destination port and transport protocol.

Table A.2: Macrotraces characteristics with pre-processing results.

Macrotrace	Flows			Packets	
	TCP		UDP		
	Complete	Ignored		Original	Filtered
User	440 k	241 k	1.1 M	118 M	10.1 M
Media&Games	11 k	4 k	16 k	81 M	2 M
Malware	392 k	466 k	979 k	33 M	26 M
IoT	39 k	79 k	50 k	5 M	2 M

different numbers of flows. Thus, I need to build a common rule to compare results.

Table A.1 summarizes the number of flows reported by each tool. I record major differences, e.g., Zeek usually identifies more flows than Tstat, even when configured with similar timeouts. This happens because of the way midstream traffic and incomplete flows are processed by the tools.

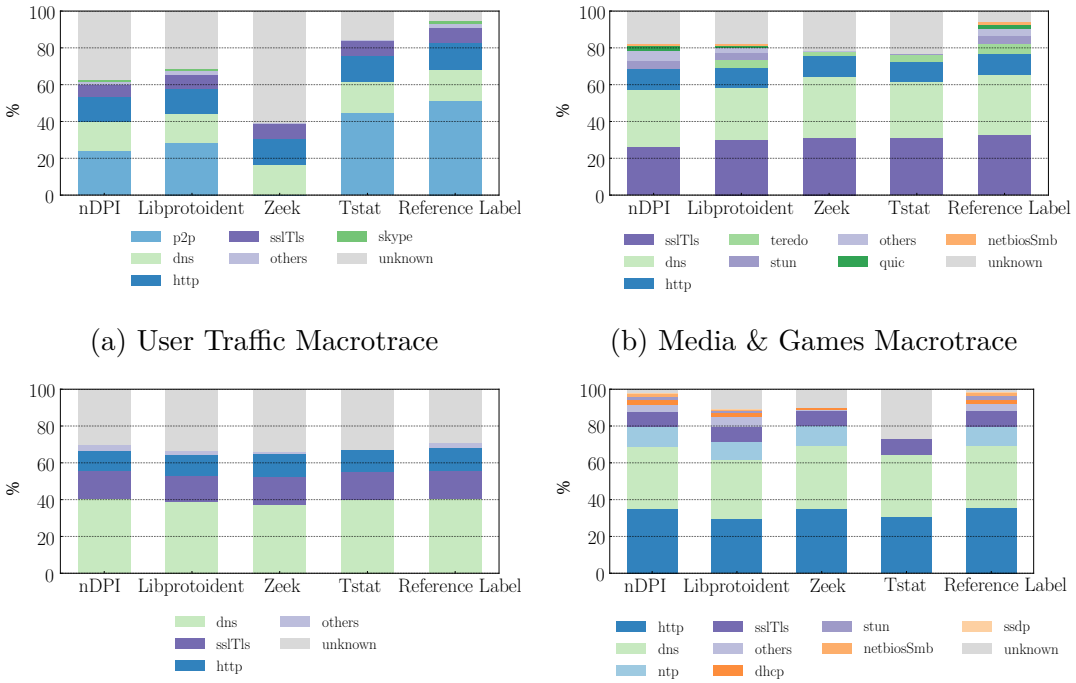
Most of the cases creating discrepancies are however not interesting for the final analysis, since they usually refer to flows that carry no payload. Indeed, a lot of flows without payload are present in particular for the Malware traces due to internet scanning traffic. These flows cannot be evaluated with DPI. As such, I perform a *pre-processing step* using Tstat as reference to keep in the final macrotraces only complete flows, i.e., UDP flows with payload and TCP flows with complete three-way handshake. All remaining flows are discarded. Whenever possible, I set the tools with similar timeout parameters for the experiments that will follow. I next normalize results ignoring the small percentage of flows that are not revealed by tools other than Tstat to avoid artifacts related to the way flow are expired or terminated. At last, I keep only the first 20 packets per flow in the final macrotraces to speed-up the analysis (see column "Filtered"). I will show later that all tested tools achieve a final protocol classification using a small number of packets per flow. As such, this pre-processing step does not impact results.

I report a summary of the final macrotraces in Table A.2. I show the number of packets and flows reported by Tstat, with the latter split as TCP and UDP. For TCP flows, I detail the number of *complete* and *ignored* flows.

In total, our final macrotraces include more than 3 M flows, and 40 M packets after all pre-processing steps are applied.

A.3.3 Matching flow labels

We need some common rules to normalize the output of the tools and compare their classifications. First, I normalize all labels, e.g., using always lower case and removing special characters. Then, I manually verify the output strings to identify possible synonyms used across tools. Table A.3 reports a subset of labels that require manual standardization. In total I manually evaluated 225 labels, replacing



(a) User Traffic Macrotrace

(b) Media & Games Macrotrace

(c) Malware Macrotrace

(d) IoT Macrotrace

Figure A.2: Percentage of labelled flows for each tool. The last bar in the plots reports percentages for our reference label.

Table A.3: Label standardization

Standardized Label	Original Label
p2p	p2p, edonkey, emule, ed2k, cacaoweb, kademlia, bittorrent, torrent
netbiosSmb	netbios, smb, smb2, nbns
krb	krb, kerberos, spnego-krb5spnego
dns	dns, llmnr, mdns
sslTls	ssl, tls
skype	skype, skypetcp
ldap	ldap, cldap
quic	quic, gquic

cases such as those in the right column of Table A.3 by a single common label (left column).

Next, I face the question on how to determine the label for each flow in absence

Table A.4: Example of flow label constistency and score.

Flow ID	Tool				Reference Label	Score
	Tstat	Libprotoident	nDPI	Zeek		
1	krb	krb	krb	krb	krb	1
2	unk	unk	unk	unk	unk	1
3	krb	unk	krb	krb	krb	0.75
4	unk	unk	krb	krb	krb	0.5
5	unk	unk	unk	krb	krb	0.25
6	unk	sip	unk	p2p	conflict	0
7	krb	krb	p2p	p2p	conflict	0

of ground truth. Indeed, the lack of ground truth has pushed most of previous works to resort to testbeds or emulated traffic that I want to avoid [156]. I thus decide to focus on the *consistency* of different tools, i.e., I assume that the most common normalized label assigned to a flow is the *reference label* for such flow, and calculate a *confidence score* for each decision. In case of conflicts, we manually verify each case.

Table A.4 reports examples of classification, along with the per-flow *confidence score*. The easiest cases happen when there is an unanimous decision towards the same protocol (e.g., Flow 1) or towards the unknown label (e.g., Flow 2). Both decisions result in a score equals to 1. When at least one tool is able to recognize the protocol, I ignore the unknown labels and pick the recognized label as reference label. Yet, our *confidence score* is lower in this case, e.g., see Flow 5. It rarely happens (e.g., Flow 6) that all tools recognize a different protocol, or there is a draw (e.g., Flow 7). Some of these cases have been solved by inspecting the source code of the DPI tools, e.g., giving preference to labels found by pattern matching over those guessed based on port numbers or other heuristics. The few cases I could not resolve are ignored, with confidence score equals to zero.

Finally, once the reference labels are defined, I calculate performance metrics for each tool. I consider the following metrics: (i) accuracy, the percentage of flows with label matching the reference; (ii) precision (per protocol), the percentage of such flows that match with the reference; and (iii) recall (per protocol), the percentage of such flows the tool has classified as the given protocol.

A.4 Results

Here, I show a summary of the identified flows per tool and I summarize the classification performance in the several scenarios. Next, I discuss the performance in terms of the number of packets required to reach a steady classification, and briefly discuss computational performance of tools.

A.4.1 Labelled flows per protocol

Fig. A.2 shows a break-down of the number of labelled flows reported by each tool. Four plots depict results for the different macrotraces. The last bar on each plot reports the percentage of flows given by the *reference label*, i.e., the label selected by the majority of tools. Each figure reports the most common lables in order of popularity.

In the User Traffic case (top-left plot), Tstat shows the best performance, reporting labels for around 85% of the flows. All the libraries recognize popular protocols (e.g., HTTP, DNS and TLS), but Libprotoident, nDPI and Zeek fail to recognize some P2P traffic, thus leaving a larger number of flows marked as unknown. Yet, notice how the number of unknown flows is small for the reference label – i.e., flows marked as unknown by Tstat are recognized by others.

In the Media & Games case – Fig. A.2b – all tools recognize close to 80% of the flows. This trace is mostly composed by HTTP, DNS and TLS traffic, which are well recognized by all tools. The reference label reports again a lower percentage of unknown than each single tool, showing potential for achieving higher classifications by merging the output of different tools.

The analysis of the Malware macrotrace – Fig. A.2c – leads to worse numbers for all cases. The percentage of labelled flows ranges from 66% to 70%. Here the presence of UDP scans towards multiple ports impact results. Manual inspection shows the presence of payload that matches the fingerprints of scan UDP attacks against certain IoT devices. None of the tools is able to identify the protocol of this malicious traffic, calling for specialized DPI approach in security use cases.

In the IoT case – Fig. A.2d – nDPI is the best performing, labelling almost all flows. Tstat is penalized by the lack of fingerprints for NTP, STUN and SSDP. All in all, most flows in this trace are labelled by at least one tool (see the reference label bar).

Finally, I evaluate the average confidence scores for different protocols. With this analysis, I aim at identifying protocols for which the tools demonstrate high consistency. Fig. A.3 shows the average scores for flows labeled with one of the top-20 protocols considering all four macrotraces. Common protocols such as TLS, HTTP and NTP are recognized with an average score higher or equal to 75% (left side of the figure). That is, such protocols are consistently identified by at least three tools on average. As I move to less popular labels, the confidence scores reduce significantly. Indeed, the score is reduced to around 25% for Netbios, QUIC and SSDP (right side of the figure). In other words, only one tool outputs a label for flows carrying these protocols, with others marking flows as unknown.

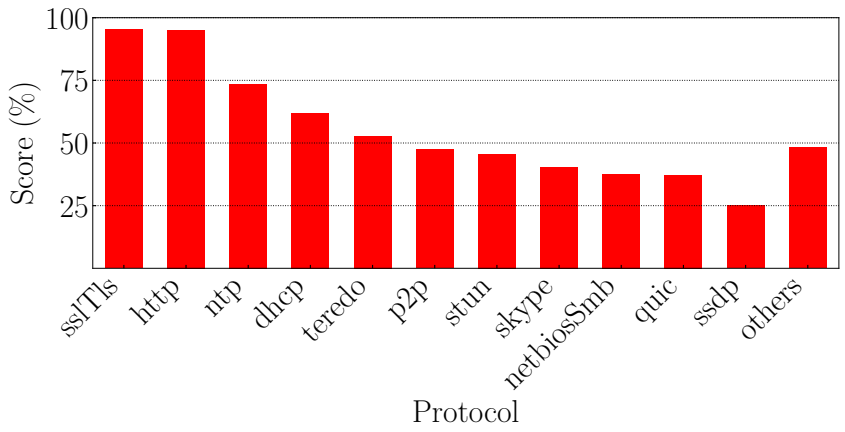


Figure A.3: Average per flow confidence score for the top reference labels.

A.4.2 Classification performance

I next quantify the percentage of flows classified by each tool as well as their classification performance in respect to the reference labels. Results are presented in Tab. A.5. I highlight in bold the best performing tool per trace and metric.

Consider the first row group in the table. It reports the percentage of labelled flows, summarizing the results presented in the previous section. As said, Tstat reports more labels for the User Traffic scenario, thanks to its abilities to spot P2P flows. nDPI instead reaches the largest percentages in the other scenarios, thanks to its capabilities to guess labels based on multiple heuristics.

Considering accuracy (second row group), the numbers are similar to those for labelled flows across all scenarios. That is, the overall accuracy (with regards to the reference labels) is driven by the percentage of unknown flows reported by each tool. Yet, some particular cases can be noticed, such as minor differences between nDPI and Libprotoident in the Media & Games Macrotrace. These minor mismatches arise from cases in which one of the tools, although capable to label the given flow, disagree with the label given by the majority. These cases are rare and indeed confirm that once tools labels a flow, the provided label is usually reliable.

Zeek wins when it comes to the average precision per protocol (third row group), almost always reaching 100%. That is, when Zeek recognizes a protocol, its label matches the reference. Yet, Zeek suffers in terms of average recall (fourth row group), due to its limited set of labels. Libprotoident, on the other hand, reaches the highest average recall per protocol in most scenarios, which can be explained by its large set of labels, with over 200 protocols. nDPI shows balanced numbers for both precision and recall per protocol. nDPI find a good number of labels (high recall) that usually match with the reference (high precision).

Table A.5: Summary of classification results.

Metric	Library	Macrotrace			
		User Traffic	Games & Media	Malware	IoT
Labelled Flows	Tstat	**0.85**	0.77	0.67	0.73
	Libprotoident	0.69	**0.86**	0.66	0.89
	nDPI	0.63	**0.86**	**0.70**	**0.98**
	Zeek	0.40	0.78	0.66	0.89
Accuracy	Tstat	**0.85**	0.77	0.67	0.73
	Libprotoident	0.69	**0.82**	0.66	0.85
	nDPI	0.62	0.79	**0.70**	**0.98**
	Zeek	0.40	0.78	0.66	0.89
Average Precision	Tstat	0.99	0.87	0.98	**1**
	Libprotoident	0.96	0.91	0.99	0.80
	nDPI	0.93	0.89	**1**	0.99
	Zeek	**1**	**0.97**	**1**	**1**
Average Recall	Tstat	0.71	0.62	**1**	1
	Libprotoident	**1**	**0.89**	**1**	0.94
	nDPI	0.82	0.78	**1**	**1**
	Zeek	0.66	0.62	0.97	0.79

A.4.3 How many packets are needed for DPI?

I analyze the performance of tools while limiting the number of packets per flow. This test has been performed by cutting off each flow after observing its n first packets *with payload*, i.e., ignoring initial TCP handshake packets. Flows composed by n or less packets with payload are kept untouched. The goal is to evaluate the number of packets needed to reach a final classification, and whether labels change as more packets are observed.

Fig. A.4 shows the resulting average accuracy among all macrotraces. Clearly results do not change when increasing the number of packets, and all tools reach an almost steady classification after just one packet. Some tools (e.g., nDPI) increase accuracy further after observing the second packet with payload, but gains are marginal. This result is particularly relevant, as DPI tools are often used for real-time identification of protocols on security applications. Note that nDPI has average accuracy slightly superior than others, with Libprotoident and Tstat coming next.

Finally, I also controlled the performance of the tools in terms of memory fingerprint and processing time. Here a general conclusion is hard to be reached, since the tools are delivered for different target scenarios. For example, the basic

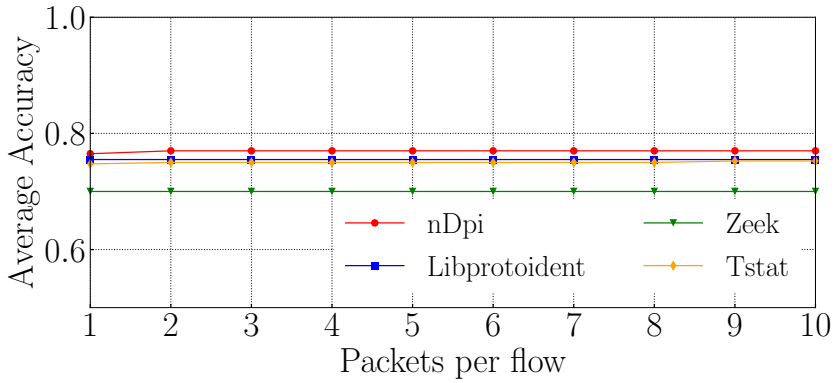


Figure A.4: Average accuracy when increasing the number of packets per flow. Tools reach a final classification already in the first packet with payload.

installation of Zeek runs as multiple processes, prepared to handle several Gbps. Libprotoident and nDPI are libraries that can also be integrated in simple demonstration programs. In our tests, all tool, but Zeek, present similar performance figures when processing a single PCAP at a time.

A.5 Conclusions

I presented an evaluation of DPI solutions in several traffic scenarios, comparing the consistency of their classifications. The tools are practically equivalent when the input traffic is composed by popular and well-known protocols (e.g., HTTP, DNS and TLS). When applied to complex scenarios, such as traffic generated by Malware scans, DPI tools struggle. I also observed discrepancies on the classification of less popular protocols, with some protocols being supported by only one of the tools. In sum, there is space for improving these DPI tools by extending their label sets. Interestingly, tools reach steady-state classification after one packet, suggesting they can be exploited in online scenarios.

Appendix B

List of publications

1. **Soro, F.**, Favale, T., Giordano, D., Vassio, L., Ben Houidi, Z., & Drago, I. (2021). The New Abnormal: Network Anomalies in the AI Era. *Communication Networks and Service Management in the Era of Artificial Intelligence and Machine Learning*, 261-288.

2. Rescio, T., Favale, T., **Soro, F.**, Mellia, M., & Drago, I. (2021). DPI Solutions in Practice: Benchmark and Comparison. In *2021 IEEE Security and Privacy Workshops (SPW)* (pp. 37-42). IEEE.

3. Trevisan, M., **Soro, F.**, Mellia, M., Drago, I., & Morla, R. (2020). Does domain name encryption increase users' privacy?. *ACM SIGCOMM Computer Communication Review, 50(3)*, 16-22.

4. Favale, T., **Soro, F.**, Trevisan, M., Drago, I., & Mellia, M. (2020). Campus traffic and e-Learning during COVID-19 pandemic. *Computer networks, 176*, 107290.

5. **Soro, F.**, Allegretta, M., Mellia, M., Drago, I., & Bertholdo, L. M. (2020). Sensing the Noise: Uncovering Communities in Darknet Traffic. In *2020 Mediterranean Communication and Computer Networking Conference (MedComNet)* (pp. 1-8). IEEE.

6. **Soro, F.**, Mellia, M., & Russo, N. (2020). Regular Pattern and Anomaly Detection on Corporate Transaction Time Series. In *EDBT/ICDT Workshops*.

7. **Soro, F.**, Drago, I., Trevisan, M., Mellia, M., Ceron, J., & Santanna, J. J. (2019). Are darknets all the same? On darknet visibility for security monitoring. In *2019 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)* (pp. 1-6). IEEE.

8. Khatouni, A. S., **Soro, F.**, & Giordano, D. (2019). A machine learning application for latency prediction in operational 4g networks. In *2019 IFIP/IEEE*

Symposium on Integrated Network and Service Management (IM) (pp. 71-74). IEEE.

9. Faroughi, A., Javidan, R., Mellia, M., Morichetta, A., **Soro, F.**, & Trevisan, M. (2018). Achieving horizontal scalability in density-based clustering for URLs. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 3841-3846). IEEE.

10. Casas, P., **Soro, F.**, Vanerio, J., Settanni, G., & D'Alconzo, A. (2017). Network security and anomaly detection with Big-DAMA, a big data analytics framework. In *2017 IEEE 6th international conference on cloud networking (CloudNet)* (pp. 1-7). IEEE.

Bibliography

[1] Davide Abati et al. “Latent space autoregression for novelty detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2019.

[2] Subil Abraham and Suku Nair. “A predictive framework for cyber security analytics using attack graphs”. In: *arXiv preprint arXiv:1502.01240* (2015).

[3] G. Aceto et al. “PortLoad: Taking the Best of Two Worlds in Traffic Classification”. In: *2010 INFOCOM IEEE Conference on Computer Communications Workshops*. 2010, pp. 1–5.

[4] ADBHoney. *Low interaction honeypot designed for Android Debug Bridge over TCP/IP*. 2021. URL: https://github.com/huuck/ADBHoney.

[5] Shikha Agrawal and Jitendra Agrawal. “Survey on anomaly detection using data mining techniques”. In: *Procedia Computer Science* 60 (2015), pp. 708–713.

[6] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. “A survey of network anomaly detection techniques”. In: *Journal of Network and Computer Applications* 60 (2016), pp. 19–31.

[7] Leman Akoglu, Hanghang Tong, and Danai Koutra. “Graph based anomaly detection and description: a survey”. In: *Data mining and knowledge discovery* 29.3 (2015), pp. 626–688.

[8] Eric Alata et al. “Lessons learned from the deployment of a high-interaction honeypot”. In: *2006 Sixth European Dependable Computing Conference*. IEEE. 2006, pp. 39–46.

[9] S. Alcock and R. Nelson. “Measuring the accuracy of open-source payload-based traffic classifiers using popular Internet applications”. In: *38th Annual IEEE Conference on Local Computer Networks - Workshops*. 2013, pp. 956–963.

[10] Shane Alcock and Richard Nelson. “Libprotoident: Traffic Classification Using Lightweight Packet Inspection”. In: (2012).

[11] Omar Alrawi et al. “The Circle Of Life: A Large-Scale Study of The IoT Malware Lifecycle”. In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 2021.

[12] M. Antonakakis et al. “Understanding the Mirai Botnet”. In: *Proceedings of the 26th USENIX Security Symposium*. USENIX Security'17. 2017, pp. 1093–1110.

[13] Tim Bai et al. “RDP-Based Lateral Movement Detection using Machine Learning”. In: *Computer Communications* 165 (2021), pp. 9–19.

[14] Michael Bailey et al. “Practical Darknet Measurement”. In: *Proc. of the CISS*. 2006, pp. 1496–1501.

[15] Michael Bailey et al. “The Internet Motion Sensor: A Distributed Blackhole Monitoring System”. In: *Proc. of the NDSS*. 2005, pp. 167–179.

[16] F. Baker, W. Harrop, and G. Armitage. *IPv4 and IPv6 Greynets*. Tech. rep. 6018. RFC Editor, 2010.

[17] Ariel Bar et al. “Identifying attack propagation patterns in honeypots using Markov chains modeling and complex networks analysis”. In: *2016 IEEE international conference on software science, technology and engineering (SWSTE)*. IEEE. 2016, pp. 28–36.

[18] Paul Barford et al. “Employing honeynets for network situational awareness”. In: *Cyber situational awareness*. Springer, 2010, pp. 71–102.

[19] Michela Becchi, Mark Franklin, and Patrick Crowley. “A workload for evaluating deep packet inspection architectures”. In: *2008 IEEE International Symposium on Workload Characterization*. IEEE. 2008, pp. 79–89.

[20] Laura Beggel, Michael Pfeiffer, and Bernd Bischl. “Robust anomaly detection in images using adversarial autoencoders”. In: *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2019.

[21] Yoshua Bengio, Aaron Courville, and Pascal Vincent. “Representation learning: A review and new perspectives”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[22] Karyn Benson et al. “Leveraging Internet Background Radiation for Opportunistic Network Analysis”. In: *Proc. of the IMC*. 2015, pp. 423–436.

[23] Paul Bergmann et al. “Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE. 2020.

[24] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal K Kalita. “Network anomaly detection: methods, systems and tools”. In: *Ieee communications surveys & tutorials* 16.1 (2013), pp. 303–336.

[25] Vincent D Blondel et al. "Fast unfolding of communities in large networks". In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008.

[26] Matt Boddy, Ben Jones, and Mark Stockley. "RDP Exposed-The Threat That's Already at Your Door". In: *Sophos, Inc, Sophos White Paper* (2019).

[27] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[28] N. Brownlee. "One-way Traffic Monitoring with Iatmon". In: *Proceedings of the 13th International Conference on Passive and Active Measurement*. PAM'12. 2012, pp. 179–188.

[29] A. Brzeczko et al. "Active Deception Model for Securing Cloud Infrastructure". In: *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2014, pp. 535–540.

[30] Tomasz Bujlow, Valentín Carela-Español, and Pere Barlet-Ros. "Independent comparison of popular DPI tools for traffic classification". In: *Computer Networks* 76 (2015), pp. 75–89.

[31] CAIDA/UCSD. *The UCSD Network Telescope*. 2021. URL: https://www.caida.org/projects/network_telescope/.

[32] Emmanuel J Candès et al. "Robust principal component analysis?" In: *Journal of the ACM (JACM)* 58.3 (2011), pp. 1–37.

[33] Pedro Casas et al. "Network security and anomaly detection with BigDAMA, a big data analytics framework". In: *2017 IEEE 6th international conference on cloud networking (CloudNet)*. IEEE. 2017, pp. 1–7.

[34] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.

[35] Ajay Chaudhary and Anjali Sardana. "Software based implementation methodologies for deep packet inspection". In: *2011 international conference on information science and applications*. IEEE. 2011, pp. 1–10.

[36] Jinghui Chen et al. "Outlier detection with autoencoder ensembles". In: *Proceedings of the SIAM international conference on data mining*. SIAM. 2017.

[37] Cisco. "Cisco annual internet report (2018–2023) white paper". In: (2020). URL: https://www.%20cisco.%20com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/whitepaper-c11-741490.%20html.

[38] Evan Cooke et al. "Toward Understanding Distributed Blackhole Placement". In: *Proc. of the WORM*. 2004, pp. 54–64.

[39] Gianpiero Costantino and Ilaria Matteucci. "CANDY CREAM-haCking infotAiNment anDroid sYstems to Command instRument clustEr via cAn data fraMe". In: *Proceedings of the IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*. IEEE. 2019, pp. 476–481.

[40] Marc Coudriau, Abdelkader Lahmadi, and Jerome Francois. "Topological analysis and visualisation of network monitoring data: Darknet case study". In: *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE. 2016, pp. 1–6.

[41] Cowrie. *SSH/Telnet Honeypot*. 2021. URL: https://github.com/cowrie/cowrie.

[42] Jakub Czyz et al. "Understanding IPv6 Internet Background Radiation". In: *Proc. of the IMC*. 2013, pp. 105–118.

[43] A. Dainotti et al. "Extracting Benefit from Harm: Using Malware Pollution to Analyze the Impact of Political and Geophysical Events on the Internet". In: *SIGCOMM Comput. Commun. Rev.* 42.1 (2012), pp. 31–39.

[44] Alberto Dainotti et al. "Analysis of Country-Wide Internet Outages Caused by Censorship". In: *IEEE/ACM Trans. Netw.* 22.6 (2014), pp. 1964–1977.

[45] Alberto Dainotti, Antonio Pescapé, and Giorgio Ventre. "A packet-level characterization of network traffic". In: *2006 11th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks*. IEEE. 2006, pp. 38–45.

[46] Alberto Dainotti et al. "Analysis of a "/0" Stealth Scan From a Botnet". In: *IEEE/ACM Trans. Netw.* 23.2 (2015), pp. 341–354.

[47] Alberto Dainotti et al. "Lost in Space: Improving Inference of IPv4 Address Space Utilization". In: *IEEE Journal on Selected Areas in Communications* 34.6 (2016), pp. 1862–1876.

[48] Emiliano De Cristofaro et al. "Paying for Likes? Understanding Facebook Like Fraud Using Honeypots". In: *Proceedings of the 2014 Conference on Internet Measurement Conference*. IMC '14. Vancouver, BC, Canada, 2014, pp. 129–136. ISBN: 9781450332132.

[49] L. Deri et al. "nDPI: Open-source High-speed Deep Packet Inspection". In: *Proceedings of the International Wireless Communications and Mobile Computing Conference*. IWCMC. 2014, pp. 617–622.

[50] Andrea Di Domenico et al. "A network analysis on cloud gaming: Stadia, GeForce Now and PSNow". In: *arXiv preprint arXiv:2012.06774* (2020).

[51] Paraskevi Dinaki. "Deep Packet Inspection: A Comparison Study Between Exact Match and Regular Expression Techniques". In: (2018).

[52] Dionaea. *Generic Low Interaction Honeypot*. 2021. URL: https://github.com/DinoTools/dionaea.

[53] Harris Drucker et al. "Support vector regression machines". In: *Advances in neural information processing systems*. 1997, pp. 155–161.

[54] Zakir Durumeric, Michael Bailey, and J. Alex Halderman. "An Internet-Wide View of Internet-Wide Scanning". In: *Proc. of the SEC*. 2014, pp. 65–78.

[55] Elias Raftopoulos et al. "How Dangerous Is Internet Scanning? A Measurement Study of the Aftermath of an Internet-Wide Scan". In: *Proc. of the TMA*. 2015, pp. 158–172.

[56] Scott Emmons et al. "Analysis of network clustering algorithms and cluster quality metrics at scale". In: *PloS one* 11.7 (2016).

[57] Claude Fachkha, Elias Bou-Harb, and Mourad Debbabi. "Inferring Distributed Reflection Denial of Service Attacks from Darknet". In: *Comput. Commun.* 62.C (2015), pp. 59–71.

[58] Claude Fachkha and Mourad Debbabi. "Darknet as a Source of Cyber Intelligence: Survey, Taxonomy, and Characterization". In: *Commun. Surveys Tuts.* 18.2 (2016), pp. 1197–1227.

[59] Jinliang Fan et al. "Prefix-Preserving IP Address Anonymization: Measurement-Based Security Evaluation and a New Cryptography-Based Scheme". In: *Comput. Netw.* 46.2 (2004), pp. 253–272.

[60] Durdu Ömer Faruk. "A hybrid neural network and ARIMA model for water quality time series prediction". In: *Engineering Applications of Artificial Intelligence* 23.4 (2010), pp. 586–594.

[61] S. Fernandes et al. "Slimming Down Deep Packet Inspection Systems". In: *IEEE INFOCOM Workshops 2009*. 2009, pp. 1–6.

[62] Santo Fortunato. "Community detection in graphs". In: *Physics reports* 486.3-5 (2010), pp. 75–174.

[63] Daniel Fraunholz et al. "Data Mining in Long-Term Honeypot Data". In: *Proceedings of the IEEE International Conference on Data Mining Workshops*. ICDMW. 2017, pp. 649–656.

[64] Luca Gioacchini et al. "DarkVec: automatic analysis of darknet traffic with word embeddings". In: *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*. 2021, pp. 76–89.

[65] Glutton. *Generic Low Interaction Honeypot*. 2021. URL: https://github.com/mushorg/glutton.

[66] Izhak Golan and Ran El-Yaniv. "Deep anomaly detection using geometric transformations". In: *Proceedings of the Advances in Neural Information Processing Systems*. NIPS. 2018.

[67] Dong Gong et al. "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection". In: *Proceedings of the IEEE International Conference on Computer Vision*. IEEE. 2019.

[68] GreyNoise. 2021. URL: https://greynoise.io/.

[69] Francesco Gringoli et al. "Gt: picking up the truth from the ground for internet traffic". In: *ACM SIGCOMM Computer Communication Review* 39.5 (2009), pp. 12–18.

[70] Wonkyu Han et al. "HoneyMix: Toward SDN-Based Intelligent Honeynet". In: *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. SDN-NFV Security'16. USA, 2016, pp. 1–6.

[71] Warren Harrop and Grenville Armitage. "Defining and Evaluating Greynets (Sparse Darknets)". In: *Proc. of the LCN*. 2005, pp. 344–350.

[72] Mahmudul Hasan et al. "Learning temporal regularity in video sequences". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE. 2016.

[73] Heralding. *Credentials Catching Honeypot*. 2021. URL: https://github.com/johnnykv/heralding.

[74] Raphael Hiesgen et al. "Spoki: Unveiling a New Wave of Scanners through a Reactive Network Telescope". In: *arXiv preprint arXiv:2110.05160* (2021).

[75] Honeynet. *The Honeynet Project*. 2021. URL: https://www.honeynet.org/.

[76] Honeytrap. *Advanced Honeypot Framework*. 2021. URL: https://github.com/honeytrap/honeytrap.

[77] Chengqiang Huang et al. "Towards experienced anomaly detector through reinforcement learning". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[78] Tomoharu Iwata and Makoto Yamada. "Multi-view anomaly detection via robust probabilistic latent variable models". In: *Proceedings of the Advances in neural information processing systems*. NIPS. 2016.

[79] Mattijs Jonker et al. "Millions of Targets Under Attack: A Macroscopic Characterization of the DoS Ecosystem". In: *Proc. of the IMC*. 2017, pp. 100–113.

[80] P. Joshi and H. Dinesha. "Survey on Identification of Malicious Activities by Monitoring Darknet Access". In: *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*. 2020, pp. 346–350.

[81] Karin Kandananond. "Electricity demand forecasting in buildings based on ARIMA and ARX models". In: *Proceedings of the 8th International Conference on Informatics, Environment, Energy and Applications*. ACM. 2019, pp. 268–271.

[82] A. R. Khakpour and A. X. Liu. "High-Speed Flow Nature Identification". In: *2009 29th IEEE International Conference on Distributed Computing Systems*. 2009, pp. 510–517.

[83] Jawad Khalife, Amjad Hajjar, and Jesús Díaz-Verdejo. "Performance of OpenDPI in identifying sampled network traffic". In: *Journal of Networks* 8.1 (2013), p. 71.

[84] Jack Kiefer et al. "Stochastic estimation of the maximum of a regression function". In: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 462–466.

[85] Hyunchul Kim et al. "Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices". In: *Proceedings of the 2008 ACM CoNEXT Conference*. CoNEXT '08. Madrid, Spain: Association for Computing Machinery, 2008. ISBN: 9781605582108.

[86] A. King et al. "A Coordinated View of the Temporal Evolution of Large-scale Internet Events". In: *Computing* 96.1 (2014), pp. 53–65.

[87] M. Kuhrer et al. "Exit from Hell? Reducing the Impact of Amplification DDoS Attacks". In: *Proceedings of the 23rd USENIX Conference on Security Symposium*. SEC'14. 2014, pp. 111–125.

[88] Atsutoshi Kumagai, Tomoharu Iwata, and Yasuhiro Fujiwara. "Transfer anomaly detection by inferring latent domain representations". In: *Proceedings of the Advances in Neural Information Processing Systems*. NIPS. 2019.

[89] Mahesh Kumar, Nitin R Patel, and Jonathan Woo. "Clustering seasonality patterns in the presence of errors". In: *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2002.

[90] Sailesh Kumar, Jonathan Turner, and John Williams. "Advanced algorithms for fast and scalable deep packet inspection". In: *2006 Symposium on Architecture For Networking And Communications Systems*. IEEE. 2006, pp. 81–92.

[91] Donghwoon Kwon et al. "A survey of deep learning-based network anomaly detection". In: *Cluster Computing* (2019), pp. 1–13.

[92] Sofiane Lagraa, Yutian Chen, and Jérôme François. "Deep mining port scans from darknet". In: *International Journal of Network Management* 29.3 (2019), e2065.

[93] Sofiane Lagraa and Jérome François. "Knowledge discovery of port scans from darknet". In: *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE. 2017, pp. 935–940.

[94] Andrea Lancichinetti and Santo Fortunato. "Community detection algorithms: a comparative analysis". In: *Physical review E* 80.5 (2009), p. 056117.

[95] Max Landauer et al. "System log clustering approaches for cyber security applications: A survey". In: *Computers & Security* 92 (2020), p. 101739.

[96] Ian XY Leung et al. "Towards real-time community detection in large networks". In: *Physical Review E* 79.6 (2009), p. 066107.

[97] Steffen Liebergeld, Matthias Lange, and Ravishankar Borgaonkar. "Cellpot: A Concept for Next Generation Cellular Network Honeypots". In: *Internet Society* (2014), pp. 1–6.

[98] Peng Lizhi et al. "Traffic labeller: collecting internet traffic samples with accurate application information". In: *China Communications* 11.1 (2014), pp. 69–78.

[99] Gabriel Maciá-Fernández et al. "UGR '16: A new dataset for the evaluation of cyclostationarity-based network IDSs". In: *Computers & Security* 73 (2018), pp. 411–424.

[100] Marco Mellia, R Lo Cigno, and Fabio Neri. "Measuring IP and TCP behavior on edge nodes with Tstat". In: *Computer Networks* 47.1 (2005), pp. 1–21.

[101] Lionel Metongnon and Ramin Sadre. "Beyond Telnet: Prevalence of IoT Protocols in Telescope and Honeypot Measurements". In: *Proc. of the WTMC*. 2018, pp. 21–26.

[102] Lihua Miao, Wei Ding, and Haiting Zhu. "Extracting Internet Background Radiation from Raw Traffic using Greynet". In: *Proc. of the ICON*. 2012, pp. 370–375.

[103] University of Michigan. *Why am I receiving connection attempts from the University of Michigan?* 2013. URL: https://cse.engin.umich.edu/about/resources/connection-attempts/.

[104] Andrew W Moore and Konstantina Papagiannaki. "Toward the accurate identification of network applications". In: *International Workshop on Passive and Active Network Measurement*. Springer. 2005, pp. 41–54.

[105] David Moore et al. "Inferring Internet Denial-of-Service Activity". In: *ACM Trans. Comput. Syst.* 24.2 (2006), pp. 115–139.

[106] David Moore et al. *Network Telescopes: Technical Report*. Tech. rep. 2004.

[107] Farnaz Moradi, Tomas Olovsson, and Philippas Tsigas. "An evaluation of community detection algorithms on large-scale email traffic". In: *International symposium on experimental algorithms*. Springer. 2012, pp. 283–294.

[108] S. Morishita et al. “Detect Me If You... Oh Wait. An Internet-Wide View of Self-Revealing Honeypots”. In: *Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. 2019, pp. 134–143.

[109] M. Nawrocki et al. “A Survey on Honeypot Software and Data Analysis”. In: *arXiv:1608.06249* (2016).

[110] Mark EJ Newman. “Modularity and community structure in networks”. In: *Proceedings of the national academy of sciences* 103.23 (2006), pp. 8577–8582.

[111] Antonio Nisticó et al. “A comparative study of RTC applications”. In: *To appear in the Proceedings of the 22nd IEEE International Symposium on Multimedia* (2020).

[112] Steven Noel et al. “Cygraph: graph-based analytics and visualization for cybersecurity”. In: *Handbook of Statistics*. Vol. 35. Elsevier, 2016, pp. 117–167.

[113] Ping-Feng Pai and Chih-Sheng Lin. “A hybrid ARIMA and support vector machines model in stock price forecasting”. In: *Omega* 33.6 (2005), pp. 497–505.

[114] Ruoming Pang et al. “Characteristics of Internet Background Radiation”. In: *Proc. of the IMC*. 2004, pp. 27–40.

[115] A Parmisano, S Garcia, and MJ Erquiaga. “A Labeled Dataset with Malicious and Benign IoT Network Traffic”. In: *Stratosphere Laboratory: Praha, Czech Republic* (2020).

[116] Vern Paxson. “Bro: a system for detecting network intruders in real-time”. In: *Computer networks* 31.23-24 (1999), pp. 2435–2463.

[117] Cynthia Phillips and Laura Painton Swiler. “A graph-based system for network-vulnerability analysis”. In: *Proceedings of the 1998 workshop on New security paradigms*. 1998, pp. 71–79.

[118] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. “Near linear time algorithm to detect community structures in large-scale networks”. In: *Physical review E* 76.3 (2007), p. 036106.

[119] Stephen Ranshous et al. “Anomaly detection in dynamic networks: a survey”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 7.3 (2015), pp. 223–247.

[120] K Hanumantha Rao et al. “Implementation of anomaly detection technique using machine learning algorithms”. In: *International Journal of Computer Science and Telecommunications* 2.3 (2011), pp. 25–31.

[121] Tommaso Rescio et al. “DPI Solutions in Practice: Benchmark and Comparison”. In: *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE. 2021, pp. 37–42.

[122] Philipp Richter and Arthur Berger. “Scanning the Scanners: Sensing the Internet from a Massively Distributed Network Telescope”. In: *Proceedings of the Internet Measurement Conference*. IMC ’19. Amsterdam, Netherlands, 2019, pp. 144–157.

[123] Jean-Pierre van Riel and Barry Irwin. “InetVis, a visual tool for network telescope traffic analysis”. In: *Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*. 2006, pp. 85–89.

[124] V. Riyadi. *Securing Mikrotik Router*. 2018. URL: https://mum.mikrotik.com/presentations/ID18/presentation_5554_1540255240.pdf.

[125] Martin Rosvall et al. “Memory in network flows and its effects on spreading dynamics and community detection”. In: *Nature communications* 5.1 (2014), pp. 1–13.

[126] R. E. Schapire. “Explaining adaboost”. In: *Empirical inference*. Springer, 2013, pp. 37–52.

[127] Thomas Schlegl et al. “f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks”. In: *Medical image analysis* 54 (2019), pp. 30–44.

[128] Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang. “Accurate, scalable in-network identification of p2p traffic using application signatures”. In: *Proceedings of the 13th international conference on World Wide Web*. 2004, pp. 512–521.

[129] Shaoqiang Wang, DongSheng Xu, and ShiLiang Yan. “Analysis and application of Wireshark in TCP/IP protocol teaching”. In: *2010 International Conference on E-Health Networking Digital Ecosystems and Technologies (EDT)*. Vol. 2. 2010, pp. 269–272.

[130] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. “Toward generating a new intrusion detection dataset and intrusion traffic characterization.” In: *ICISSP*. 2018, pp. 108–116.

[131] Chaofan Shen and Leijun Huang. “On detection accuracy of L7-filter and OpenDPI”. In: *2012 Third International Conference on Networking and Distributed Computing*. IEEE. 2012, pp. 119–123.

[132] Arunan Sivanathan et al. “Classifying IoT devices in smart environments using network traffic characteristics”. In: *IEEE Transactions on Mobile Computing* 18.8 (2018), pp. 1745–1759.

[133] SNARE/TANNER. *Web Application Honeypot Sensor*. 2021. URL: http://mushmush.org/.

[134] Pavol Sokol, Jakub Míšek, and Martin Husák. "Honeypots and Honeynets: Issues of Privacy". In: *EURASIP Journal on Information Security* 2017.1 (2017), p. 4. ISSN: 1687-417X.

[135] F. Soro et al. "Sensing the Noise: Uncovering Communities in Darknet Traffic". In: *Proceedings of the Mediterranean Communication and Computer Networking Conference*. MedComNet. 2020, pp. 1–8.

[136] Francesca Soro, Marco Mellia, and Nicolo Russo. "Regular Pattern and Anomaly Detection on Corporate Transaction Time Series." In: *EDBT/ICDT Workshops*. 2020.

[137] Francesca Soro et al. "Are Darknets All The Same? On Darknet Visibility for Security Monitoring". In: *2019 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE. 2019, pp. 1–6.

[138] Francesca Soro et al. "The New Abnormal: Network Anomalies in the AI Era". In: *Communication Networks and Service Management in the Era of Artificial Intelligence and Machine Learning* (2021).

[139] Stuart Staniford et al. "The Top Speed of Flash Worms". In: *Proc. of the WORM*. 2004.

[140] Géza Szabó et al. "On the validation of traffic classification algorithms". In: *International Conference on Passive and Active Network Measurement*. Springer. 2008, pp. 72–81.

[141] Jay Thom, Yash Shah, and Shamik Sengupta. "Correlation of Cyber Threat Intelligence Data Across Global Honeypots". In: *Proceedings of the IEEE 11th Annual Computing and Communication Workshop and Conference*. CCWC. 2021, pp. 0766–0772.

[142] Christof Ferreira Torres, Mathis Steichen, and Radu State. "The Art of The Scam: Demystifying Honeypots in Ethereum Smart Contracts". In: *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA, 2019, pp. 1591–1607.

[143] TPot. *The All In One Honeypot Platform*. 2021. URL: https://github.com/telekom-security/tpotce.

[144] M. Trevisan et al. "Traffic Analysis with Off-the-Shelf Hardware: Challenges and Lessons Learned". In: *IEEE Commun. Mag.* 55.3 (2017), pp. 163–169.

[145] Fang-Mei Tseng et al. "Fuzzy ARIMA model for forecasting the foreign exchange market". In: *Fuzzy sets and systems* 118.1 (2001), pp. 9–19.

[146] Twisted. *Event-driven Networking Engine Written in Python*. 2021. URL: https://twistedmatrix.com/trac/.

[147] Himani Tyagi and Rajendra Kumar. "Attack and Anomaly Detection in IoT Networks Using Supervised Machine Learning Approaches." In: *Rev. d'Intelligence Artif.* 35.1 (2021), pp. 11–21.

[148] Vladimir Vapnik et al. "Support vector method for function approximation, regression estimation and signal processing". In: *Advances in neural information processing systems*. 1997, pp. 281–287.

[149] Alexander Vetterl and Richard Clayton. "Bitter Harvest: Systematically Fingerprinting Low- and Medium-interaction Honeypots at Internet Scale". In: *Proceedings of the 12th USENIX Workshop on Offensive Technologies (WOOT 18)*. Baltimore, MD, USA: USENIX Association, 2018.

[150] VirusTotal. 2021. URL: https://www.virustotal.com/.

[151] Paul Wagenseller, Feng Wang, and Weili Wu. "Size matters: A comparative analysis of community detection algorithms". In: *IEEE Transactions on Computational Social Systems* 5.4 (2018), pp. 951–960.

[152] Matthias Wählisch et al. "First insights from a mobile honeypot". In: *ACM SIGCOMM Computer Communication Review* 42.4 (2012), pp. 305–306.

[153] ZiHan Wang et al. "Automatically Traceback RDP-based Targeted Ransomware Attacks". In: *Wireless Communications and Mobile Computing* 2018 (2018).

[154] Gavin Watson. *A comparison of header and deep packet features when detecting network intrusions*. Tech. rep. 2018.

[155] Eric Wustrow et al. "Internet Background Radiation Revisited". In: *Proc. of the IMC*. 2010, pp. 62–74.

[156] Jinghua Yan. "A survey of traffic classification validation and ground truth collection". In: *2018 8th International Conference on Electronics Information and Emergency Communication (ICEIEC)*. IEEE. 2018, pp. 255–259.

[157] Vinod Yegneswaran, Paul Barford, and Vern Paxson. "Using Honeynets for Internet Situational Awareness". In: *In Proceedings of the Fourth Workshop on Hot Topics in Networks*. HotNets. 2005, pp. 17–22.

[158] Vinod Yegneswaran, Paul Barford, and Dave Plonka. "On the design and use of Internet sinks for network abuse monitoring". In: *International Workshop on Recent Advances in Intrusion Detection*. Springer. 2004, pp. 146–165.

[159] Chuxu Zhang et al. "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 1409–1416.

[160] Yiru Zhao et al. “Spatio-temporal autoencoder for video anomaly detection”. In: *Proceedings of the 25th ACM international conference on Multimedia*. ACM. 2017.

[161] Chong Zhou and Randy C Paffenroth. “Anomaly detection with robust deep autoencoders”. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2017.

[162] Nur Zincir-Heywood, Marco Mellia, and Yixin Diao. “Overview of Artificial Intelligence and Machine Learning”. In: *Communication Networks and Service Management in the Era of Artificial Intelligence and Machine Learning* (2021).

[163] Bo Zong et al. “Deep autoencoding gaussian mixture model for unsupervised anomaly detection”. In: *Proceedings of the International Conference on Learning Representations*. 2018.

[164] Tanja Zseby et al. “The Day after Patch Tuesday: Effects Observable in IP Darkspace Traffic”. In: *Proc. of the PAM*. 2013, pp. 273–275.

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was pdfLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi=scudo`. This class is available in every up-to-date and complete TEX-system installation.