Strengthening Privacy and Cybersecurity through Anonymization and Big Data

Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

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

Strengthening Privacy and Cybersecurity through Anonymization and Big Data

By

Thomas Favale

******

Supervisor(s):
Prof. Marco Mellia, Supervisor
Prof. Danilo Giordano, Co-Supervisor

Doctoral Examination Committee:
Prof. Chadi Barakat, Referee, Inria, Université Côte d'Azur
Prof. Jose Luis Garcia Dorado, Referee, Universidad Autónoma de Madrid
Prof. Giancarlo Ruffo, Università di Torino
Prof. Guido Marchetto, Politecnico di Torino
Prof. Robert Renè Maria Birke, Università di Torino

Politecnico di Torino
December 2022

Declaration

I hereby declare that, the contents and organization 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.

Thomas Favale
December 2022

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

Acknowledgements

`Thomas Favale, rapporto missione.`
`Dopo tre anni di confino nel Politecnico di Torino, sono pronto per concludere la mia missione segreta nello spazio inesplorato.`

Oggi sono qui a presentare un resoconto di tutto il mio lavoro. Un lavoro durato tre anni. Tre anni vissuti in un perpetuo moto oscillatorio tra gioie e dolori, soddisfazioni e delusioni, vittorie e sconfitte, come un pendolo. Un pendolo che non si arresta mai e scandisce il tempo che trascorre inesorabile. Ma è arrivato il momento di fermarlo un attimo per ricordare che le missioni più difficili non possono essere completate se non grazie all'aiuto di tutti coloro che credono in te. Io posso ritenermi fortunato per aver incontrato ed avuto al mio fianco persone fantastiche e che vorrei citare qui di seguito così da imprimere per sempre quanto siano state importanti per la mia vita.

Come potrei non iniziare parlando dei miei genitori: Caterina e Pierpaolo. Non voglio certo essere banale o scontato nelle mie parole, ma certo è che senza di loro non sarei qui a scrivere questo rapporto missione. Instancabili come pochi, lontani nello spazio ma vicini nel tempo, sono sempre stati pronti ad aiutarmi nei momenti difficili ed incoraggiarmi ad esprimere il miglio. Voglio, quindi, dedicare loro questo mio lavoro come ringraziamento per tutto ciò che hanno fatto e per dare loro occasione di rispolverare l'inglese e leggere qualche articolo scientifico di qualità.

E poi c'è mio fratello Gianluca, compagno di una vita, prima di giochi e poi di studio, con cui ho condiviso praticamente tutto e che a modo suo è sempre stato al mio fianco. Un po' mi mancano quegli sguardi di intesa e sorrisi da tonti, usati come mezzi di comunicazione per le futilità di tutti i giorni, come mi manca anche il fatto di non averlo intorno dopo così tanti anni, ma colgo l'occasione per augurargli il meglio per gli anni avvenire.

Ringrazio poi Valeria per essere sempre stata presente e pronta ad aiutarmi in ogni situazione della mia vita, piccola o grande che fosse. Probabilmente la persona più positiva sulla faccia della Terra che mi ha donato il sorriso anche nei momenti più difficili. Con questa menzione vorrei mandarle un in bocca al lupo, affinché possa realizzare tutti i suoi sogni, piccoli e grandi, così da vivere una vita felice ed indimenticabile.

Vorrei estendere i miei ringraziamenti coloro che in famiglia mi sono davvero stati sempre vicino e che mi hanno incoraggiato nel corso degli anni.

In aggiunta, vorrei estendere i miei ringraziamenti a tutti i miei colleghi del Centro Interdipartimentale SmartData@Polito, in particolar modo al mio supervisore Prof. Marco Mellia, che per me ha sempre rappresentato un faro nella tempesta: è riuscito a farmi amare quel che faccio, instillandomi una passione irrefrenabile per la ricerca di ciò che è ignoto, ma anche di ciò che abbiamo sotto il naso e che non riusciamo a vedere. Nulla è banale o scontato e a volte bisogna scavare nel profondo ed in maniera innovativa per far emergere ciò che veramente stiamo cercando.

Amici di una vita e, come tali, indimenticati ed indimenticabili, Gianluca e Marco. Due persone che augurerei a chiunque di incontrare. Sicuramente coloro che più mi sono rimasti nel cuore durante la mia vita da studente universitario, ma presenti anche dopo, con cui ho condiviso, alcuni tra i momenti più belli della mia vita. Spero che indipendentemente da cosa riserverà la vita, avrò modo di continuare a collezionare con loro esperienze indimenticabili.

Infine, sembrerà strano agli occhi di tutti, ma voglio davvero ringraziare Torino. Dannata all'inizio, ma che ho imparato ad amare negli anni per la storia, per la bellezza e per le persone che ci vivono o ci hanno vissuto. Un gioiello che porterò per sempre nel mio cuore per tutte le esperienze ed avventure, positive e negative, che ho vissuto, perché questo non sia un addio, ma un semplice arrivedérci.

Ora, però, vorrei ritagliarmi un piccolo spazio anche per me, così da voltarmi verso il me del passato e rivivere questi anni in un battito di ciglia per ricordare al me del futuro di combattare con la stessa determinazione e di realizzare l'impossibile.

Verso l'infinito e oltre!

Partners Acknowledgements

I would like to thank all the partners who contributed to the realization of these research projects and the related results. In particular:

- SmartData@PoliTO center (Italy)
- Huawei R&D Center (France)
- EU Project PIMCity
- European Union's Horizon 2020 research and innovation program
- Politecnico di Torino AreaIT staff (Italy)
- Consortium GARR, Gruppo per l'Armonizzazione della Rete della Ricerca (Italy)
- Intesa Sanpaolo Anti Financial Crime Digital Hub (Italy)
- Telefonica Research Center (Spain)
- University of Würzburg (Germany)

Abstract

The problem of Cybersecurity, together with Privacy are becoming increasingly pervasive and tangled especially in recent years with the relentless expansion of the Internet: it is becoming more and more important in our lives and it is the foundation for every business. In this scenario, malicious entities develop very rapidly and create new and increasingly sophisticated threats. Since their success could cause various catastrophes for each of us, it is important to be able to monitor the network and analyze the traffic captured through passive sensors such as Darknets, or active systems, as Honeypots. In the first case we are able to observe unwanted traffic (often referred to as Internet Background Radiation), generally produced by heavy-hitter sources. Moreover, coupling this tool together with active Honeypots, helps to further enrich the visibility on malicious events. Honeypots, on the other hand, are able to reply to unsollicited requests, providing a broader knowledge on the threat scenario, by engaging the potential attacker.

However, this study scenario clashes with the need to guarantee user privacy: capturing traffic anywhere on the network can involve packets generated by not malicious users, therefore identity disclosure could be a serious problem. For all these reasons, in this thesis I demonstrate how it is possible to perform network monitoring safely. As a first step, I propose an anonymization system called α-MON which is able to capture network packets at high speeds (multiple Gb/s) and apply a desired level of obfuscation for potentially sensitive information. Here I refer not only to the network addresses in the headers, but also to the payload: many protocols are still completely in clear (e.g. DNS), or, if encrypted, they expose the name of the reference service (TLS). It is important to identify them and consequently act to hide the aforementioned names contained in the payloads. The z-anonymity algorithm, on which α-MON is based, is also innovative and has its foundations in the k-anonymity, but

working on a streaming fashion: the goal is still to hide uncommon identifiers that can easily be traced back to specific users, leaving the most common unchanged. I show that this approach is promising and that the use of such tools does not degrade the quality of the data acquired by studying network traffic during the Covid-19 pandemic. Here I provide a completely new view of the changes in traffic patterns from the point of view of a Campus Network, underlining how the network has been resilient and able to withstand a sudden increase in traffic caused by users forced to work from home.

In this thesis, in addition to the privacy issue, I provide my contribution for the analysis of malicious traffic. In particular, I conducted studies on the performance of Deep Packet Inspection (DPI) systems with the aim of identifying the most suitable to operate in a streaming fashion. This is a fundamental point for creating a smart honeypot capable of understanding the protocol autonomously and actively responding consistently to attackers who do not necessarily use standard ports (for protocols) to perform their activities. I called this tool DPIPot. It is able to reveal attack patterns that would otherwise remain hidden because the most widespread and documented Honeypots tend to look only for the standard ports of the protocol that they are able to handle. Subsequently, I deployed DPIPot and a series of further responders within an address space dedicated to Darknets, demonstrating an increase in attacking sources. This is interesting, since the infrastructure has allowed me to analyze all incoming traffic horizontally: my goal is to demonstrate that attackers are able not only to hit single services with considerable insistence, but also to interact with different protocols by cycling through all those left open at their disposal. I show that the purposes can be extremely varied, starting from Crawlers for the discovery of services to brute force attackers who try to login with usernames and passwords obtained from published leaks.

In conclusion, the objective of this thesis is to demonstrate that despite the recent enhancements regarding privacy, network monitoring is still possible without loss of data quality. In addition, I demonstrate how the use of mixed passive and active probe infrastructures allows the discovery of new and more visible attack patterns.

Contents

List of Figures **xiv**

List of Tables **xviii**

1 Introduction and motivations **1**

2 Network Traffic Anonymization for Passive Monitoring **6**

2.1 Introduction 6

2.1.1 My Contribution 7

2.2 Related Work 8

2.3 z-anonymity 10

2.3.1 *z-anonymity* definitions 11

2.4 α-MON design 13

2.4.1 Deployment scenario and requirements 13

2.4.2 Packet ingestion and forwarding design 14

2.4.3 Anonymization modules 15

2.4.4 *z-anonymity* implementation and data structures 17

2.4.5 Auxiliary data structures 19

2.5 The impact of *z-anonymity* on traffic measurements 19

2.5.1 Anonymized volumes 20

2.5.2 Impact of *z-anonymity* on traffic accounting 21

2.5.3 Load on α-MON data structures . . . 24
2.6 Performance Evaluation . . . 25
2.6.1 Testbed and dataset . . . 25
2.6.2 Horizontal scalability . . . 27
2.6.3 Benchmark with other workloads . . . 28
2.6.4 α-MON sub-module performance . . . 30
2.6.5 *z-anonymity* on other protocol fields . . . 31
2.6.6 Tuning of the *z-anonymity* data structure . . . 32
2.7 Discussion on the z-anonymity approach . . . 33
2.8 Conclusion . . . 34

3 Generalizing z-anonymity as Zero-Delay Anonymization for Data Streams 36

3.1 Introduction . . . 36
3.1.1 My Contribution . . . 37
3.2 z-anonymity . . . 39
3.2.1 The z-anonymity approach . . . 39
3.2.2 Implementation and complexity . . . 40
3.3 Modelling z-anonymity and k-anonymity . . . 41
3.3.1 User and attribute popularity model . . . 42
3.3.2 Applying *z-anonymity* . . . 43
3.3.3 The attacker point of view . . . 44
3.3.4 Getting to k-anonymity . . . 44
3.4 Comparing z-anonymity and k-anonymity . . . 46
3.4.1 The impact of the attribute rank . . . 48
3.4.2 The impact of A . . . 48
3.4.3 The impact of z . . . 50

3.4.4 The impact of U . . . 50
3.4.5 The impact of N . . . 51
3.5 A practical use case: the visits to websites . . . 52
3.6 Limitations and future work . . . 55
3.7 Conclusion . . . 56

4 Analysis of Campus Traffic during COVID-19 Pandemic 58
4.1 Introduction . . . 58
4.1.1 My Contribution . . . 60
4.2 Related Work . . . 61
4.3 Datasets and methodology . . . 62
4.3.1 Passive traces . . . 63
4.3.2 Application logs . . . 63
4.3.3 Virtual classrooms . . . 64
4.3.4 Security monitors . . . 65
4.3.5 Ethics . . . 65
4.4 Impact on campus traffic and remote working solutions . . . 66
4.4.1 Aggregate traffic volume . . . 66
4.4.2 Smart working adoption . . . 67
4.4.3 Remote collaboration . . . 69
4.5 Online teaching . . . 69
4.5.1 Audience . . . 70
4.5.2 Network workload . . . 71
4.6 Network performance . . . 74
4.6.1 Internet Service Providers breakdown . . . 74
4.6.2 Geographical characteristics . . . 76
4.7 Security events and unsolicited traffic . . . 78

4.8 Conclusions 80

5 DPI Performances Evaluation for Live Applications 83

5.1 Introduction 83

5.1.1 My Contribution 84

5.2 Related Work 85

5.3 Datasets and Methodology 86

5.3.1 Selection of DPI Tools 86

5.3.2 Selection and pre-processing of traces 87

5.3.3 Matching flow labels 90

5.4 Results 91

5.4.1 Labelled flows per protocol 92

5.4.2 Classification performance 94

5.4.3 How many packets are needed for DPI? 94

5.5 Conclusions 96

6 Augmenting Darknet Capabilities through Smart Honeypots 97

6.1 Introduction 97

6.1.1 My Contribution 98

6.2 Related work 100

6.2.1 Darknet research and infrastructure 100

6.2.2 Honeypot systems and analysis 101

6.3 Methodology and datasets 102

6.3.1 Infrastructure 102

6.3.2 Data capture and processing 104

6.3.3 First experiment: Deployments and categories 105

6.3.4 Second experiment: Deploying and removing responders 106

6.3.5 Ethics . . . 107
6.4 Macroscopic changes in traffic . . . 108
6.4.1 Breakdown per flow stage . . . 108
6.4.2 Temporal evolution . . . 111
6.5 Ports and senders . . . 111
6.5.1 Changes on probed ports . . . 112
6.5.2 Changes on traffic senders . . . 114
6.6 Amplification of service-specific deployments . . . 117
6.6.1 Service amplification . . . 117
6.6.2 Targeted services and *Side-Scans* . . . 119
6.6.3 DPIpot additional visibility . . . 121
6.7 Darkening and enlightening networks . . . 124
6.7.1 From light to darkness, and back . . . 125
6.7.2 Disturbing the neighbours . . . 126
6.8 Conclusion . . . 128

7 Exploring Application Level Attackers' Interactions 130
7.1 Introduction . . . 130
7.1.1 My Contribution . . . 131
7.2 Related Work . . . 132
7.3 Methodology and Dataset . . . 132
7.4 Honeypot Traffic Patterns . . . 135
7.4.1 Evolution over Time . . . 135
7.4.2 Popular Sources and Countries . . . 137
7.4.3 Vertical vs. Horizontal Activity . . . 138
7.5 Brute-Force Attacks . . . 141
7.5.1 Known vs. Unknown Passwords . . . 141

7.5.2 Origins of Brute-Force Attacks 143
7.6 Conclusions 145

8 Conclusions **146**

References **149**

Appendix A Publication, collaborations and projects **165**
A.1 Publications 165
A.2 Research collaborations 167
A.3 Projects 167
A.3.1 α-MON 167
A.3.2 DPI-Pot 168

List of Figures

1.1 Topology of the data collection system and cybersecurity environment. 2

2.1 *z-anonymity* concept. Three users access the FQDN `private.com` over time. When less than $z = 3$ unique users' are seen in the past ΔT, requests must be anonymized. 12

2.2 Deployment scenario: α-MON anonymizes the traffic coming from a span port or an optical splitter and forwards it to different legacy monitors. 13

2.3 Data structure used to handle quasi-identifiers. 17

2.4 Fraction of traffic obfuscated by *z-anonymity* with different values of z and ΔT. z-anonymized field: FQDN 20

2.5 Fraction of traffic obfuscated by *z-anonymity* with different values of z and ΔT. z-anonymized field: Second-level domain. . 20

2.6 Per-service volume measured from an z-anonymized trace. . . . 22

2.7 Per-service flows measured on the original and on a $(z = 10)$-anonymized trace. 23

2.8 Domains known by *z-anonymity* with different ΔT, in relation with the memory needed by α-MON. 24

2.9 Performance with four input and four output interfaces using the ISP-FULL trace. 27

2.10 Performance with different traces and consumer numbers. 28

2.11 Percentage of time spent on the most impacting modules. 30

2.12 Performance when applying *z-anonymity* on IP addresses and FQDNs (ISP-FULL trace). 31

2.13 Analysis of the behaviour of Hash(QuasiID) collision list. Lines represent different hash table sizes. 32

3.1 The probability p_a^Y for a user to publish attribute a in Δt, according to its rank. 47

3.2 he impact of A on p_{k-anon}, considering both different k and z values. 48

3.3 The impact of z on p_{k-anon} for different k values. 50

3.4 The impact of U and z on p_{k-anon} for different k values ($z = 20$). 51

3.5 The impact of observation time N on p_{k-anon}, considering both different k and z values. 52

3.6 The probability p_a^Y to publish attribute a in a $\Delta t = 1\,hour$, according to its rank, as estimated from the users' navigation data ($U = 9\,670$, $A = 27\,482$). 54

3.7 The relation between *z-anonymity* and k-anonymity in the users' navigation data ($U = 9\,670$, $A = 27\,482$). 55

4.1 Variations in the mobility patterns in Italy between February and April 2020. Colors mark the days in which new Ministerial Decrees introduced mobility restrictions. Source: https://www.apple.com/covid19/mobility. 59

4.2 Traffic from three Italian Universities before and after the lockdown. 67

4.3 Daily accesses to in-house PoliTO smart working systems. . . . 67

4.4 Daily activity on PoliTO Microsoft Teams systems. 68

4.5 Daily number of virtual classrooms and connected students (faculties and students). 70

4.6 Access pattern to the teaching services. 72

4.7 Characteristics of virtual classroom sessions. 73

4.8 Access to teaching servers per ISP. Only flows larger than 10 MB are considered. 75

4.9 Access to teaching material per Italian region. Only flows larger than 10 MB are considered. 76

4.10 RTT and throughput distributions (flows larger than 10 MB) per Italian region. 77

4.11 Variations in the number of events reported by the university firewall between February and March 2020. 79

4.12 Variation in the darknet traffic between January and March 2020. 80

5.1 Testing methodology. 86

5.2 Percentage of labelled flows for each tool. The last bar in the plots reports percentages for my reference label. 92

5.3 Average per flow confidence score for the top reference labels. . . 93

5.4 Average accuracy when increasing the number of packets per flow. Tools reach a final classification already in the first packet with payload. 95

6.1 Infrastructure overview. The infrastructure is divided from no interaction (Darknet) to the highest level of interaction (DPIpot). 102

6.2 Flows reaching different deployments. Numbers inside the left plot mark the increase in respect to *Darknet Ext.* 109

6.3 Number of flows per deployment. Each bar in the figure reports numbers for a single IP address of the several deployments. . . . 110

6.4 Temporal evolution of the number of flows. 112

6.5 Number of flows per destination port for the four deployments. The presence of different active responders changes the observed traffic per port. 113

6.6 Fraction of flows per sender IP address. 114

6.7 Activity pattern of top-1000 sender IP addresses. Each row corresponds to a sender IP address. 115

6.8 Top-100 senders vs. destination port. Addresses are sorted numerically. 116

6.9 Amplification factor for the most targeted ports. 117

6.10 Amplification factor for selected deployments. β marks cases of *Side-Scans*. 120

6.11 Flows percentages on top-10 ports for DPIpot and different L7 protocols. 122

6.12 Flows per port (RDP). Zoom on first 300 ports in inner axis. . . 123

6.13 Effects of removing and adding active responders in darknets. . 124

6.14 Example of host discovery patterns observed in the infrastructure. 125

6.15 Jaccard Index among aggressive senders targeting each IP address. 127

7.1 Number of IP sources per service for each day of observation. . . 134

7.2 Evolution on number of IP sources for four honeypots. 135

7.3 Activity of IP sources probing at application level around the period I shutdown the honeypots. 135

7.4 Source overlap among the top-12 services. 139

7.5 Attackers' activity. 139

7.6 Attacks with passwords from known/unknown sources. 142

7.7 Unknown and known passwords for multiple honeypots. 143

7.8 Percentage of password attempts per origin country. 144

List of Tables

2.1 Comparison of α-MON and alternatives. 9

2.2 Packet traces. 25

3.1 Terminology used to model *z-anonymity* and *k*-anonymity. . . . 46

3.2 The default values used for the model. 47

5.1 Flows exported by the different tools before the pre-processing. . 89

5.2 Macrotraces characteristics with pre-processing results. 89

5.3 Label standardization 90

5.4 Example of flow label constistency and score. 90

5.5 Summary of classification results. 96

6.1 Deployments and protocols. Each row refers to the traffic of 8 IP addresses during my first capture period (from 15/04/2021 to 16/06/2021). For direct comparison, I report numbers only for the 8 first addresses in Darknet Ext. 105

6.2 IP allocation in the fresh deployment of active responders. . . . 107

6.3 Amplification for L4-Responders and L7-Responders. Cases in which no amplification is observed are marked with a hyphen. . 119

6.4 Top-5 protocols recognized in DPIpot. 121

7.1 Honeypot services and amount of traffic seen in my deployment. 133

7.2 Distribution of traffic per service coming from some known sources. 134

7.3 Percentage L7 requests per country considering all honeypots. . 138

Chapter 1

Introduction and motivations

The problem of *Cybersecurity* has become increasingly pervasive and cumbersome especially in recent years with the relentless expansion of the Internet for accessing news, buying goods, developing businesses, building social relations and so on. Currently there are 5,382 million users (about 67.8% of the world population [1]) who have at least one device capable of surfing the Internet. In addition, we must take into account the rapid expansion of the world of the *Internet of Things* (IoT) which in 2021 counts 330 million connected devices [2] that contribute for instance to the monitoring of the territory and the simplification of everyday life. These numbers suggest how the nodes potentially subject to cyber-attacks are increasing and are getting closer to the people. It is therefore evident that the dangers deriving from cyber-attacks are no longer a problem of the *few*: if initially the targets were represented by large entities such as political or financial institutions [3], it is now appropriate for each of us to be able to have adequate protections.

These problems have recently worsened due to the advent of the *Covid-19* pandemic which represented an important challenge not only in the health sector (hospitals, pharmaceutical industries, etc.), but also in the IT one: following the report by China (31 December 2019), the Italian Government, for instance, proclaimed a state of emergency and implemented the first measures to contain the contagion, favoring the use of remote working (more commonly known as *Smart-Working*). This challenge has severely tested the resilience of network equipment due to the sudden increase in active users [4]. Given this

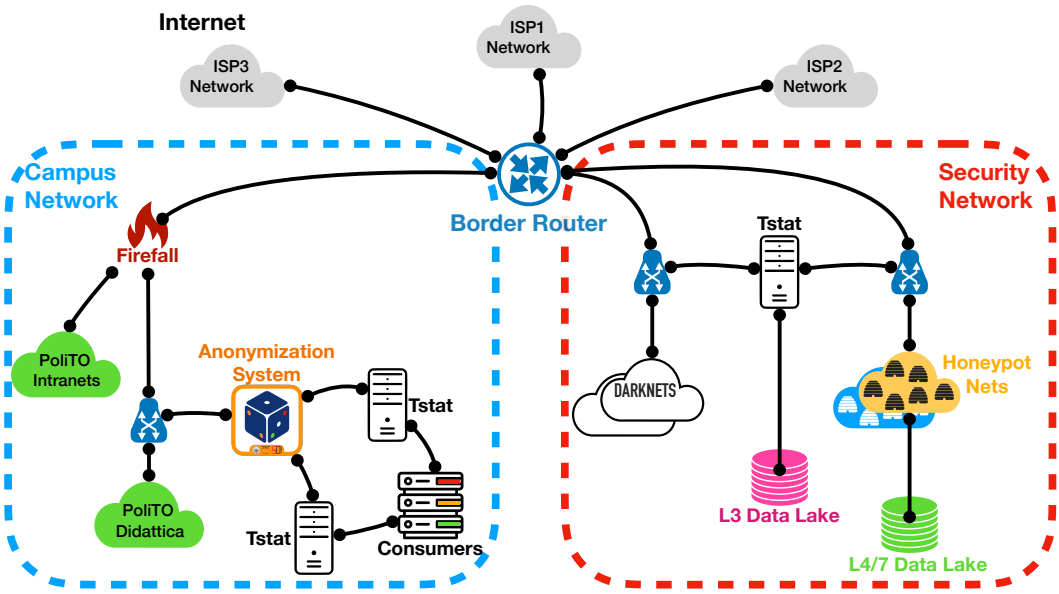


Fig. 1.1 Topology of the data collection system and cybersecurity environment.

emergency situation, *Politecnico di Torino*, as other Universities, is investing even more resources to provide adequate training to all its students. In this sense, an infrastructure has been built from scratch, in order to provide lessons interactively and remotely, reaching students in all corners of the world. But, like any new infrastructure, it needs to be monitored in order to ensure its proper functioning and guarantee security standards. Therefore it needs a monitoring system that displays all the necessary information, without compromising users' privacy: in other words, it is important to guarantee the quality of the data, together with the property of non-re-identification, in accordance with European policies [5]. To accomplish this complex task, I have developed a high performing software system called α-MON, described in Ch. 2 and Ch. 3, that executes live traffic anonymization at high speed, to cope with the network packets flowing in the campus backbone. Its integration in the *Campus Network* architecture is depicted in Fig. 1.1. All production systems are protected by firewalls, i.e., internetwork gateways that limit data communication traffic to and from *Politecnico di Torino Campus Network* to protect its resources from external threats. Systems to be monitored are located in *PoliTO Didattica* Network. In order to capture valuable traffic, a network splitter has been deployed: it is capable of replicate a single input signal into multiple outputs, making it possible to provide traffic to the anonymizer. α-MON is appropriately

configured to apply different levels of anonymization to the incoming traffic according to the downstream consumers, providing the datasets analyzed in Ch. 4, in which I will expose the impact of *Covid-19* pandemic from the point of view of a Campus Network, giving a view of the current trends in the global network.

As a side effect of this pandemic, correlated to the extensive use of *Smart-Working*, is the surge in cybercrime activity. As described before, more connected devices are deployed, more attack points attackers can gain: according to [6], in 2021, its damages amounted to $6 trillion per year, $500 billion per month, $115.4 billion per week, $16.4 billion per day, $684.9 million per hour, $11.4 million per minute, and $190,000 per second, covering mobile devices and apps [7], video games [8] and IoT devices [9, 10], to name but a few. To improve network security, it is important to study its on-going evolution, gathering information in the most targeted way possible and investigating behaviors and events in order to find effective countermeasures and keep our knowledge updated. With this in mind, I have build a secure and stable environment in which to capture data from the network. This process can be applied in two ways:

- *passive monitoring* is the packet capture by placing passive probes like Tstat [11] [1] to monitor already present network traffic, as done in Ch. 6.
- *active monitoring* is the injection of *artificial* network traffic through *Honeypots* [12] (i.e. active sensors that obtain information about attacks by answering unsolicited traffic). I will investigate on this in Ch. 7;

Since they offer different levels of visibility, they can be placed together in the same environment and used in synergy to take advantage of both solutions as depicted in the red box of Fig. 1.1. This infrastructure is capable of capturing a large amount of data in the form of network traces and logs at the application level, calling for a big data approach: it requires the use of scalable software and hardware such as cluster servers running Apache Hadoop [13] and Spark [14]. Their processing speed is critical for producing results in a short amount of time, thanks to their distributed architecture that allows to increase the available resources by relying on different servers (called *Nodes*) of the cluster

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

of computers. These aspects play a fundamental role, allowing me to analyze months of traffic efficiently and in an *horizontal* fashion.

In this thesis I will provide a study on two aspects of the cybersecurity world, namely *privacy* and *attackers behavior*. The first objective is to provide an innovative solution to anonymize network traffic in an efficient way, preventing users re-identification in a live environment and showing how minimal is the information loss. This is possible thanks to the development of a new algorithm and an anonymisation tool, which can be easily integrated into already existing passive traffic capture infrastructures. Secondly, this thesis wants to set the basis to the development of an advanced network security and monitoring system, capable of engage attackers by mimicing real services and answering "*intelligently*" to probes. This passes through the development of a smart Honeypot system based on Deep Packet Inspection, capable of engaging an attacker, regardless of the protocol or port used. Data captured from a system like this will be fundamental for future studies and to enhance the fight against cybersecurity threats.

This thesis is structured as follows: Ch. 2 introduces the problem of privacy in the context of traffic monitoring. I presentα-MON, a flexible and modular tool to anonymize network packets in a streaming fashion, with zero delay. It is able to capture network packets, anonymize them in real-time, and immediately send them to consumers for data capture and future analysis. Next in Ch. 3 I will generalize the z-anonymity algorithm, applied practically in α-MON in a real environment. I will demonstrate that it can be used in fields that differs from network packets anonymization. Here I will factually show how efficient this approach is for data collection and analysis, thanks to its ability to hide just the necessary information that can lead to users re-identification and preserving quality of the data. I have then analyzed anonymous data provided by such a system in Ch. 4. Here I will show the impact of *Covid-19* pandemic on *Politecnico di Torino* teaching systems, to observe and characterize a trend for the current state of use of the global network. From Ch. 5 I will move my focus on the behavior of attackers on the global network. Here I will present a study on the effectiveness of state of the art Deep Packet Inspection tools. This is a preliminary work to introduce the development of DPIPot in Ch. 6, an horizontal honeypot that identifies protocols on the fly, thanks to DPI, on any port to show that such an interactive responders increase the value of

darknet data, uncovering patterns otherwise unseen. In Ch. 7 I will look up at the application layer, analyzing logs generated by the honeypots. This allows me to filter out noise and backscattering from the equation and highlight the activities carried out by attackers who have tried to interact with the simulated system, also revealing and analyzing brute force attacks with the insertion of access credentials obtained from data leaks. Finally in Ch. 8 I will draw some considerations and conclusions.

Chapter 2

Network Traffic Anonymization for Passive Monitoring

To ensure user privacy in the context of network traffic capture and measurement systems, it is important to handle carefully sensitive information that may be hidden in network packets. To this end, in this chapter I propose a solution by presenting *α-MON: Traffic Anonymizer for Passive Monitoring*, published in the journal *IEEE Transaction on Network and Service Management* [15], which is an extension of my preliminary work [16], presented at *2020 32nd International Teletraffic Congress (ITC 32)*.

2.1 Introduction

Passive measurements collected from networks are fundamental to the well-functioning of the Internet. They are widely used to support applications such as cyber-security and traffic management [17, 18]. Packets flowing on network links are either saved as full-packet traces or processed on-the-fly to generate traffic summaries. Network packets, however, carry sensitive information about users. For example, HTTP, TLS and DNS traffic exposes names of services contacted by users, which in turn can be used to build users' profiles [19, 20]. Network measurements thus may expose privacy-sensitive information and shall be performed with care to avoid threatening users' privacy [21]. New privacy regulations (e.g., GDPR [5]) aim at protecting users' privacy by imposing strict

rules when handling sensitive information. They provide the interested parties rights and assign powers to the regulators to enforce these rights. Network measurements must be treated in the light of these regulations, and technology must guarantee that sensitive information is not collected unless needed.

The solution to these problems has been anonymization – i.e., replacing sensitive values by obfuscated copies. Anonymization is usually done in a per-field fashion, since different network protocol fields represent different privacy threats. Client IP addresses are *identifiers*, i.e., they allow one to identify the users (devices) generating traffic immediately. As such, they must always be obfuscated. The classic approach is CryptoPAN [22], a method that replaces IP addresses with pseudo-encrypted copies while maintaining the network prefixes. Other protocol fields, while not carrying identifiers, still pose risks as they may help user reidentification, thus acting as *quasi-identifiers*. Server IP addresses and server names (e.g., in HTTP or TLS) can be quasi-identifiers. They give hints about users' interests and, in some cases, allow user reidentification. Quasi-identifiers, therefore, shall be obfuscated too.

Replacing all identifiers and quasi-identifiers in traffic measurements with obfuscated copies reduces the value of the traces substantially. Taking again server names as an example, popular names (e.g., `www.facebook.com` or `www.google.com`) bring little information to uncover any specific user. Yet, associating traffic to particular servers is instrumental, e.g., for network management, accounting and dimensioning.

Anonymization techniques like *k-anonymity* [23] can handle quasi-identifiers – i.e., obfuscating values that allow user re-identification. However, these approaches work with *batches* of data and are impractical with high-dimensional datasets, like, e.g., the set of websites users access. In our scenario, packets arrive at very high speeds and must be processed and forwarded online with minimum delay. Storing traces for posterior offline anonymization is not a viable option.

2.1.1 My Contribution

In this chapter, I present α-MON, a flexible and modular tool to anonymize network packets in a streaming fashion, with zero delay. α-MON acts as an

anonymization device. It receives packets from the network, anonymizes them in real-time, and immediately outputs packets to multiple consumers – e.g., security monitors or passive meters.

α-MON follows a novel approach to anonymize packets on-the-fly. To this end, I introduce *z-anonymity*, a mechanism to hide infrequent field values (like unpopular server names) from the traffic. When observing a value in a data stream, *z-anonymization* removes it if less than z users share the value in the past ΔT time interval. Performing *z-anonymity* online requires ingenuity, and α-MON implements a scalable and parallel solution for this. I show that *z-anonymity* introduces minimal errors on volumetric traffic measurements, such as the estimation of traffic share of popular web services and websites.

I evaluate α-MON performance on Common-Off-The-Shelf (COTS) hardware with traces collected from operational networks. I show that: (i) α-MON helps to protect sensitive data via *z-anonymity*, preventing the disclosure of field values associated with fewer than z users; (ii) α-MON allows most information that would be obfuscated by strict per-field anonymization to be exported, thus generating richer traces than alternatives; (iii) α-MON scales to tens of Gbit/s with zero packet loss using few cores. In pessimistic scenarios, it easily achieves several Gbit/s too; (iv) α-MON introduces minimal errors on common measurement scenarios, e.g., allowing accurate accounting of the heavy-hitters' traffic. Finally, α-MON is publicly available as an open-source project.[1]

2.2 Related Work

As said, passive network monitoring threatens users' privacy [24]. Because of that, we witness significant efforts to prevent information leakage from the network, and these efforts have been mostly centered around the deployment of encryption [25, 26]. For example, all newest web protocols by the time of writing (e.g., QUIC and HTTP/2) are built to run seamlessly over TLS. These initiatives reduce the amount of information exposed during the monitoring [27]. However, users' privacy can still be exposed in certain fields of Internet protocols. Server IP addresses and FQDNs are two prominent examples, which may leak the sites visited by users. As such, those must be considered quasi-identifiers.

[1] Project is available on github.com

Table 2.1 Comparison of α-MON and alternatives.

	α-MON	ONTAS [34]	TCPdPriv [35]	TCPurify [36]
HW Implementation		✓		
SW Implementation	✓		✓	✓
Online Anon.	✓	✓	✓	✓
Stateful Anon.	✓			
L2	✓	✓		
L3	✓	✓	✓	✓
L4	✓		✓	
L5-7	✓			

Recent initiatives aim at encrypting plain-text FQDNs seen in traffic, e.g., encrypting DNS [28] and Server Name Indications (SNIs) in TLS [29]. However, not all users will adopt these technologies soon. In any case, those who monitor the network for legitimate reasons must also protect the users' privacy, as mandated by regulations [5].

Several works propose techniques to anonymize traffic by obfuscating fields of protocol headers. The goal is to allow accurate network monitoring without threatening users' privacy. We can roughly group these techniques into (i) address anonymization and (ii) payload anonymization.

Address Anonymization The simplest approach to achieve anonymization of IP addresses is the truncation of addresses. Everything, but the first n bits of the addresses (typically 8, 16 or 24), are set to zero. Truncation only partly mitigates the problem, as it is still possible to determine the subnet or the organization the truncated addresses belong to. More sophisticated techniques propose a prefix-preserving pseudo-anonymization, in which addresses are completely shuffled, but preserving the structure of subnets [30–32]. Crypto-PAN is perhaps the most popular prefix-preserving algorithms for IP addresses anonymization [33, 22]. The mappings between the original and anonymized addresses are determined by a passphrase and a symmetric block cipher. Here I rely on Crypto-PAN for IP address anonymization. Finally, in 2020 Kim *et al.* propose ONTAS [34], a flexible traffic anonymizer implemented directly in PISA-based programmable switches, which achieves high speed while anonymizing IP and MAC addresses.

Payload Anonymization Payload anonymization is more complex, as personal information may leak from different and complex protocols. Anonymization tools like `TCPdPriv` [35] and `TCPurify` [36] truncate TCP and UDP payloads, to remove all information contained in application layer protocols. This simple "reveal nothing" policy may lead to poor measurements. Other works propose sophisticated frameworks to handle specific application-level protocols. The authors of [37] remove sensitive information without affecting the payload. Packets are reconstructed into data stream flows, and application-level parsers modify the data streams as specified by a policy written in a high-level language. They provide limited anonymization primitives (constant substitution, sequential numbering, hashing, prefix-preserving, and adding random noise), forcing the user to write her own functions. The authors of [38] propose a programmable anonymization tool based on BPF filters, allowing the user to choose different actions according to the received protocol (IP, TCP, UDP, ICMP, HTTP or FTP).

Differently from these approaches, I explicitly target an operational deployment, in which anonymization must be achieved in real-time at tens of Gbit/s. Inspired by *k-anonymity*, I design a modular and flexible architecture to support *z-anonymity*. I focus on scalability and employ state-of-the-art packet capture techniques to make the system deployable on high-speed networks. Table 2.1 compares the features of α-MON against the three closest previous proposals described above, namely ONTAS [34], `TCPdPriv` [35] and `TCPurify` [36], highlighting the novel capabilities of α-MON.

2.3 *z*-anonymity

The drastic increase in the rate at which personal data are collected has pushed researchers to propose techniques to anonymize data. The goal of anonymization is to avoid disclosing personal information without compromising the utility of datasets. The seminal work of Samarati *et al.* proposes the *k-anonymity* property [23, 39]. It aims at preventing the reidentification of individuals or the extraction of sensitive information about them by ensuring that at least k individuals share the same properties in the dataset. k-anonymity

has been extended with the *l-diversity* [40] and *t-closeness* [41] ideas, which I will discuss in Section 2.7.

k-anonymity however does not scale [42] and cannot be implemented with minimal delay. With α-MON I want to achieve *some* level of anonymity already during data collection, by hiding the most sensitive information observed in network measurements. This goal calls for highly scalable and zero-delay solutions. We lie in a scenario where anonymization must be performed in real-time and must scale up to multi-Gbit/s streams. Streaming approaches for anonymity [43, 44] load the incoming records in a structure and release anonymized data in batches, which is impractical with high-speed network traffic, given the large speeds of the input streams. In sum, I cannot assume to have the whole dataset, or a large subset of the data, at disposal for anonymization.

2.3.1 *z-anonymity* definitions

Here, I propose a novel concept of anonymity. I call it *z-anonymity*. It targets real-time, online processing, with minimum latency. In the following, I provide a formal definition. I assume that users are identified by an *identifier*. The most common identifier in network traces are client IP addresses[2].

Quasi-identifiers are attributes whose values must be controlled, as they may help to re-identify users. In this case, quasi-identifiers are fields present in protocol headers and payload that may be associated with a small group of users. Examples include specific server IP addresses and server names present in payloads (e.g., in DNS) and user-agent strings (e.g., in HTTP requests). For instance, an attacker could leverage a user's interest in a particular website to re-identify her and, in turn, all her traffic. *z-anonymity* aims at obfuscating rare values of quasi-identifiers in real-time, preventing these privacy attacks. I introduce the definition of *z-private quasi-identifier*.

Definition 1. A z-private quasi-identifier is a value observed at time t that is associated with less than z users in the past ΔT time interval.

[2] *z-anonymity* can handle any protocol field as an identifier.

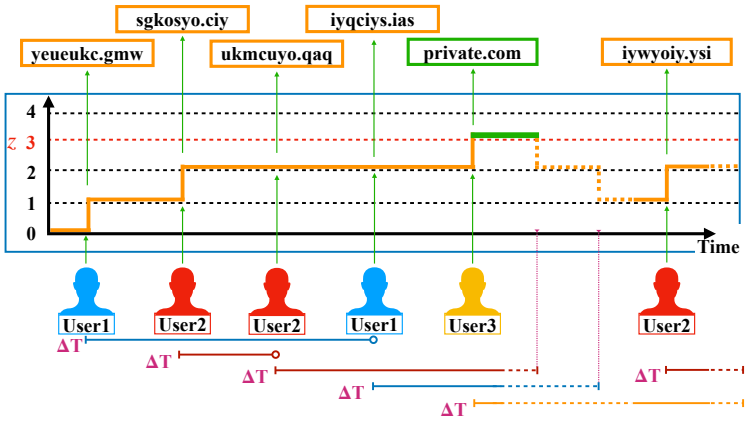


Fig. 2.1 *z-anonymity* concept. Three users access the FQDN `private.com` over time. When less than $z = 3$ unique users' are seen in the past ΔT, requests must be anonymized.

If the anonymized dataset hides z-private quasi-identifiers, it achieves z-*anonymity*.

Definition 2. A stream of packets is $z-$*anonymized* if all z-private quasi-identifiers are obfuscated, given z and ΔT.

In other words, if a quasi-identifier has been observed by at most *z-1* users in ΔT, we obfuscate it. By adjusting parameters z and ΔT, it is possible to regulate the trade-off between data utility and privacy. Indeed, a large z results in the majority of values to be anonymized, while a small z allows rare values to be exposed. ΔT regulates the memory of the system.

I exemplify the idea of *z-anonymity* in Figure 2.1. Here the quasi-identifiers are the Fully Qualified Domain Names (FQDNs) found in packet payloads, which refer to websites and web services. Suppose different users access the FQDN `private.com`. Let $z = 3$. The first four accesses shall be obfuscated as only two users accessed it up to then. When we observe User3's request, there are 3 users that have accessed `private.com` in the past ΔT. Thus, I allow User3's request to pass without anonymization. Notice that exposing `private.com` does not uncover User3, as attackers cannot know who the other two users are. After some time, User2 accesses the domain again. The previous entry for User1 is no longer in the current ΔT window, and `private.com` is anonymized again.

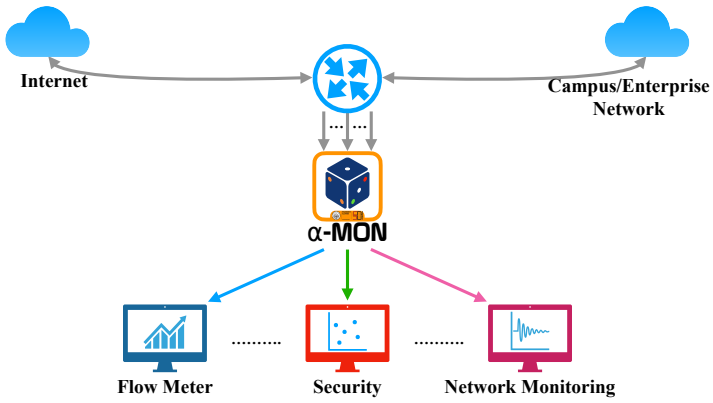


Fig. 2.2 Deployment scenario: α-MON anonymizes the traffic coming from a span port or an optical splitter and forwards it to different legacy monitors.

Clearly, in the above example, popular websites and services would be accessed by several users, and their names would not be anonymized. Rare FQDNs that could bring specific information about users would likely be anonymized. In Section 2.5, I provide a thorough analysis of how *z-anonymity* impacts the accuracy of traffic measurements, while in Chapter 3, I provide a generalization of this anonymization model.

2.4 α-MON design

I now describe α-MON, covering requirements, design choices, and implementation, with a special focus on the data structures used to achieve *z-anonymity* at high-speeds.

2.4.1 Deployment scenario and requirements

Figure 2.2 shows the deployment scenario. α-MON operates as a classic network monitor, receiving packets from one or multiple network cards, either using span ports or optical splitters. To allow legacy applications to coexist, α-MON is deployed in front of them and forward anonymized packets to multiple consumers. Compatible with best-practices for privacy, α-MON performs different anonymization according to the consumer, thus passing on the minimal information required by each legacy application. For example, a security

monitor may need traffic with little modification, while a passive meter used for traffic accounting can operate with less information.

α-MON must be flexible and support a rich set of functionalities. It shall satisfy the following requirements:

1. It must achieve *z-anonymity* to hide private quasi-identifiers with customizable z and ΔT;
2. It must support a flexible set of anonymization policies, covering all protocol layers;
3. It must be scalable and deployable in high-speed links, handling multiple tens of Gbit/s with no packet loss;
4. It must support multiple legacy applications with different anonymization requirements.

2.4.2 Packet ingestion and forwarding design

α-MON runs on a COTS server and receives packets from several network interfaces. For efficiency, I implement it in C language. For packet capture, I rely on the Data Plane Development Kit (DPDK) [45], a set of libraries and drivers for fast packet processing. α-MON follows a multi-threaded design and can take advantage of all cores available in a server. I use the architecture proposed in a previous work [46], in which the incoming packets are load-balanced to different threads – one per CPU core – using the Receive Side Scaling (RSS) feature of modern network cards. Each network interface implements load-balancing algorithms so that incoming packets are spread to multiple queues based on hash functions. This mechanism allows fast and scalable load balancing in hardware and avoids wasting CPU resources.

Some of the α-MON anonymization capabilities require stateful per-flow processing and mandate data structures to keep track of TCP and UDP flow status.[3] To avoid expensive synchronizations, network interfaces load-balance packets in a consistent per-flow fashion. In other words, packets belonging to

[3] I define a flow by the usual 5-tuple.

the same flow are always processed by the same thread. I reach this goal by instrumenting the network interface with a specific RSS hash seed [47].

Each thread receives a fraction of the overall traffic. According to custom-defined configurations, packets are replicated, their payloads anonymized, and, finally, forwarded to output interface(s) connected to the legacy monitors. To avoid concurrent access to network interfaces, α-MON sets up a transmitting queue dedicated to each thread on each network interface, again using the DPDK functionalities. Traditional techniques for increasing system robustness to failures can be applied to α-MON. For example, it can be run in multiple machines for increasing reliability, as soon as traffic is steered accordingly (i.e., consistently sending packets of a flow to the same α-MON instance). In case of very critical setups, it would be possible to replicate two identical α-MON deployments by using multiple span ports or optical splitters.

2.4.3 Anonymization modules

I design α-MON to be modular and flexible. As such, the anonymization functions build a processing pipeline. This approach eases the configuration of anonymization policies and allows new modules to be integrated into the system with little effort. α-MON supports multiple configurations, which differ, e.g., for encryption keys and anonymization pipeline. α-MON takes care of making copies of packets and performs the desired steps on each pipeline before forwarding packets to a consumer. This design allows deployments in which different consumers require different anonymization policies, e.g., security monitors receive original packets, while passive monitors receive fully-anonymized packets.

Currently, α-MON implements the following modules to search and anonymize identifiers and quasi-identifiers contained in the traffic:

Layers 5-7 The key novelty of α-MON resides in the mechanisms for handling quasi-identifiers in application-layer protocols. α-MON implements a classification engine based on Deep Packet Inspection to identify popular protocols. In its current implementation, α-MON supports quasi-identifiers contained in TLS, DNS, and HTTP protocols. In particular, α-MON can apply *z-anonymity*

on the found FQDNs, which are deleted from packets in case they are not z-private. α-MON can also apply *z-anonymity* on second-level domains – i.e., the FQDNs truncated after the top-level domain.[4] In this modality, α-MON releases the FQDN if not z-private. In case it is z-private, it checks if the second-level domain is not, and, in case, α-MON truncates the FQDN to the second-level domain. Similarly, any field of protocol headers could be subjected to *z-anonymity*, with customized z and ΔT parameters. Alternatively, the fields can be obfuscated by default (i.e., treated as an identifier).

Layer 4 α-MON keeps a table to track TCP and UDP flows, allowing per-flow anonymization policies. Tracking flows is fundamental for consistent layer 5-7 anonymization. α-MON currently does not modify L4-headers, but one could easily implement a mechanism for obfuscating potentially sensitive L4 information (e.g., rarely used TCP options).

Layer 3 α-MON considers client IP addresses as identifiers and anonymize them using the CryptoPAN algorithm [33, 22]. CryptoPAN encryption keys can be static or randomly rotated at fixed time intervals. α-MON allows the administrator to restrict the addresses that undergo anonymization to specific subnets, e.g., targeting only IP addresses of clients in the administered network. It supports IPv4 and IPv6. IP addresses that are not identifiers (e.g., server IP addresses) can be treated as quasi-identifiers and undergo *z-anonymity*.

Layer 2 α-MON supports the removal of MAC addresses. Alternatively, as routers generally modify MAC addresses once they forward the packets, α-MON can store a timestamp in place of the MAC headers. This mechanism allows consumers to get timestamps of the moment packets entered α-MON, thus increasing the precision.

Finally, α-MON implements a default policy to completely drop the payload of specific/unknown protocols at any layer – e.g., forwarding only anonymized L3 or L4 headers to consumers while removing L5-7 payloads.

[4] For example `private.example.com` becomes `example.com`.

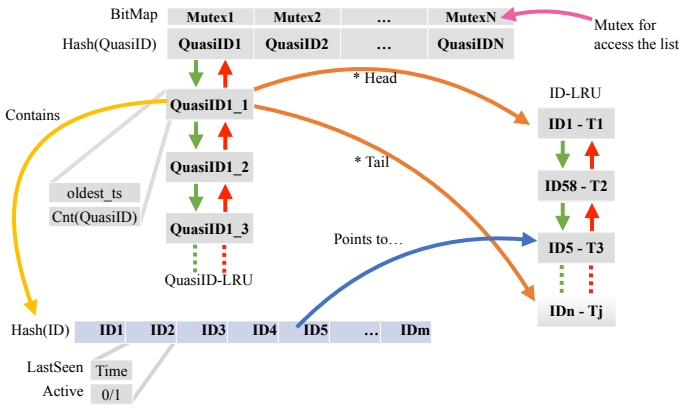


Fig. 2.3 Data structure used to handle quasi-identifiers.

2.4.4 *z-anonymity* implementation and data structures

I now describe the data structures used to implement traffic anonymization at tens of Gbit/s. To reach high speeds, it is necessary to carefully design suitable data structures that avoid expensive global synchronizations. I focus on the most challenging data structures.

α-MON includes a dedicated module for *z-anonymity*. When processing a packet from a user identified by ID and containing the quasi-identifier $QuasiID$, α-MON must decide whether to keep $QuasiID$ or hide it. The decision is based on the counter *Cnt(QuasiID)* of users sharing the $QuasiID$ in the time window ΔT.

To keep track of these counters, I rely on the specifically designed data structure depicted by Figure 2.3. As *z-anonymity* must globally count the number of users sharing each $QuasiID$, the data structure must be shared between all threads. Therefore, α-MON needs to handle concurrent accesses, which is a potentially expensive operation. Its core is composed of a shared hash table *Hash(QuasiID)*, in which each bucket is protected by a Mutex lock to handle concurrent accesses. A list handles hash collisions, organized as a Least Recently Used (LRU) structure for efficiency – *QuasiID-LRU* in the figure. Each entry in the LRU contains the information related to a quasi-identifier value ($QuasiID$). Beside metadata, it contains a second LRU, the *ID-LRU* list, that stores the ordered set of users sharing the $QuasiID$, along

with the timestamp of respective last occurrence. This *ID-LRU* is instrumental for purging those IDs whose occurrences happened more than ΔT time ago.

The metadata for $QuasiID$ contains pointers to both head and tail of the ID-LRU (illustrated by orange arrows), the oldest timestamp at which $QuasiID$ has been observed and the counter of unique IDs currently active. A second inner hash table guarantees $O(1)$ access to ID-LRU elements (illustrated by blue arrows) using the ID as key in *Hash(ID)*.

When a α-MON thread has to decide whether to anonymize or not the quasi-identifier value $QuasiID$, it first accesses the hash table *Hash(Quasi-ID)*. If $QuasiID$ is empty, the corresponding entry is created; otherwise, α-MON looks for $QuasiID$ through the collision list. Once found (or newly created), α-MON updates the QuasiID-LRU of the collision list, moving the current item to the top. Next, it updates the corresponding metadata for the $QuasiID$. Specifically, α-MON checks if the user ID is already listed among those that share $QuasiID$ in the past ΔT window. If such ID is present, its timestamp is updated to the current time. If not, the new ID is added to the ID-LRU, and the counter *Cnt(QuasiID)* of users sharing $QuasiID$ is increased. α-MON also goes through the $ID-LRU$ and deletes IDs older than ΔT. The *Cnt(QuasiID)* is decreased consequently.

At last, α-MON decides whether to anonymize $QuasiID$ based on the counter of the number of active users. If it is smaller than z, α-MON replaces the quasi-identifier value with random bytes.

Note that it is needed to purge from the data structure those entries older than ΔT. When accessing a *Hash(QuasiID)* bucket, α-MON expurges the expired entries in the *QuasiID-LRU* with a controllable probability P. This scheme limits extensive cleaning operations at each access. When cleaning *QuasiID-LRU*, α-MON controls the $ID-LRU$ for IDs older than ΔT. Again, the *Cnt(QuasiID)* is decreased consequently. If the counter indicates that the current $QuasiID$ is no longer in use, α-MON deletes it altogether.

Note that α-MON can perform most operations in $O(1)$ for each packet, thanks to the two hash tables used to access quasi-identifier values and per-identifier counters. This design allows high processing speeds as I will show in Section 2.6.2.

2.4.5 Auxiliary data structures

α-MON implements an efficient structure to support per-flow management. This structure is instrumental for applying consistent anonymization decisions based on flow state – e.g., removing the payload of specific protocols (e.g., HTTP) or parsing application layer protocols whose fields are split across multiple packets. The data structure for active flows follows the same ideas used by the authors of [11]. It builds on a hash-based data structure that provides $O(1)$ accesses to the per-flow metadata.

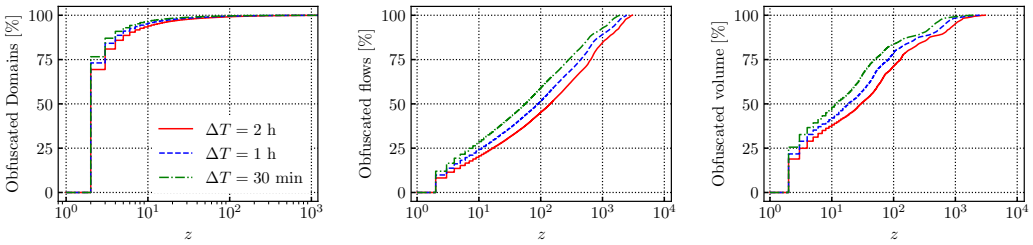
Given the current packet, the dedicated module performs a search in the flow table to verify the action performed previously: if the first packet of the flow has been subject to anonymization, the current one follows the same procedure. In the case of a new stream, α-MON creates the appropriate flow entry and checks how to anonymize the packet. The module includes a lazy garbage collection system for expired flows, similar to the one used in the *z-anonymity* module.

Finally, α-MON implements caching to speed up the anonymization of IDs, e.g., maintaining a per-thread cache of the anonymized IP addresses computed by CryptoPAN. In Section 2.6, I show that this design allows α-MON to scale to several Gbit/s of traffic.

2.5 The impact of *z-anonymity* on traffic measurements

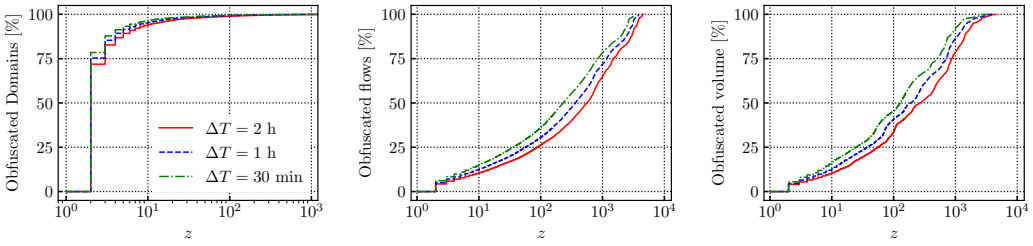
I now quantify to what extent the traffic anonymized with the *z-anonymity* is useful to provide accurate network measurements. The aim is to understand how common traffic analyses are affected when run behind an α-MON deployment. To this end, I perform a case study in which users (identified by client IP addresses) contact several hosts characterised by FQDNs (and second-level domains), which are considered quasi-identifiers.

I use a traffic trace collected on an operational network including more than 8 000 users who generate several millions of packets per second of traffic. The trace is three-days long, during which the users contacted 135 k (45 k) FQDNs (second-level domains). The FQDNs are present in TLS, DNS, and HTTP



(a) Domains (b) Flows (c) Traffic volume

Fig. 2.4 Fraction of traffic obfuscated by *z-anonymity* with different values of z and ΔT. z-anonymized field: FQDN



(a) Domains (b) Flows (c) Traffic volume

Fig. 2.5 Fraction of traffic obfuscated by *z-anonymity* with different values of z and ΔT. z-anonymized field: Second-level domain.

headers. I apply *z-anonymity* directly on FQDNs or on the corresponding second-level domains. Client IP addresses are used as identifiers. For these analyses, I implement an offline version of *z-anonymity* to process the traces, obtain statistics and show how these statistics vary with different parameters.

2.5.1 Anonymized volumes

I first analyze the fraction of traffic that *z-anonymity* would obfuscate when considering different values for z and ΔT. I show results in Figure 2.4 for the case of *z-anonymity* on FQDNs and in Figure 2.5 for second-level domains. First, consider the fraction of FQDNs that *z-anonymity* would obfuscate in Figure 2.4a. Notice that $z = 2$ already causes $\approx 75\%$ of the FQDNs to be obfuscated. When $z = 10$, the fraction increases to 90%. ΔT has a small overall impact. Similar considerations hold for the case of second-level domains (Figure 2.5a). Here, on the one hand, the coarser data granularity makes it

more likely for a domain to pass *z-anonymity*. However, I find a smaller number of quasi-identifiers (45 k instead of 135 k), which balances the picture, allowing a similar share of domains to pass *z-anonymity*.

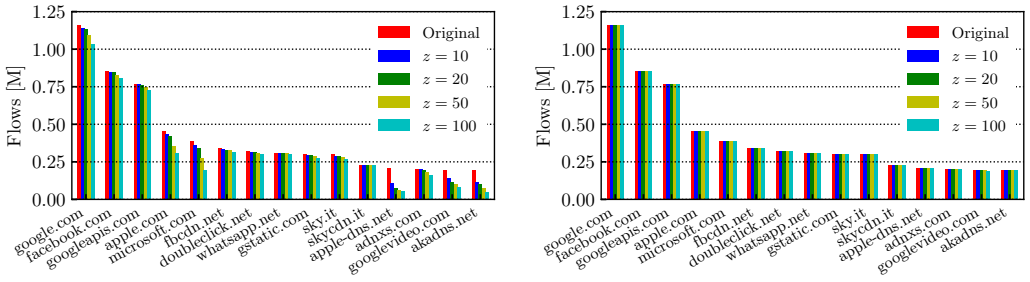
Different is the picture if I consider the number of flows (Figure 2.4b) and the byte-wise volume (Figure 2.4c) carried by flows for which the FQDN gets obfuscated. With $z = 2$, *z-anonymity* obfuscates the FQDN in only 10% of the flows, which account for $\approx 25\%$ of the traffic volume. Most of the obfuscated FQDNs are used by CDNs and include digits or random strings in the sub-domains. Taking instead the second-level domains as quasi-identifiers reduces the percentage of obfuscated bytes to negligible numbers for $z = 2$. This is caused by the nature of Internet traffic, where the majority of flows are directed towards a limited set of services [48].

The impact of a large ΔT is more pronounced for high values of z, allowing a larger number of flows to avoid obfuscation. For example, if I set $z = 100$, a $\Delta T = 30min$ results in 60% of obfuscated flows; this fraction decreases to 52 (46)% if I set $\Delta T = 1h$ ($2h$). If I consider second-level domains (Figure 2.5b and Figure 2.5c), it is possible to observe a similar picture. Considering the byte-wise volume, notice how the fraction of obfuscated traffic decreases, mainly due to the aggregation of CDN nodes and randomly generated domains to a single quasi-identifier.

In a nutshell, popular domains that carry little sensitive information are responsible for the majority of traffic. Letting their names in clear poses little threats for privacy, while still being very important for increasing visibility of network monitors. *z-anonymity* obfuscates the vast majority of FQDNs or second-level names that carry little traffic while allowing the popular names to be monitored.

2.5.2 Impact of *z-anonymity* on traffic accounting

As a use case on accounting, I study how *z-anonymity* changes the traffic volume measured for the most popular services on the network. I assume that services can be identified by their second-level domains. In Figure 2.6a I show the number of flows for the top-15 services as measured on the original trace



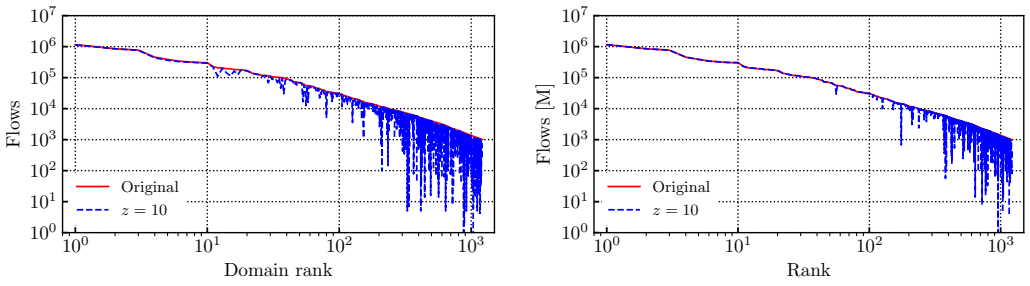
(a) *z-anonymity* on FQDN (b) *z-anonymity* on second-level domain

Fig. 2.6 Per-service volume measured from an z-anonymized trace.

and after *z-anonymity* with different values of z. In this experiment, ΔT is fixed to 1 hour, and *z-anonymity* is applied to the FQDN.

It is no surprise that the most popular service is *google.com*, followed by common services/platforms such as Facebook and Apple. The red bar represents the values measured on the original trace that I use as a baseline. If the traffic undergoes *z-anonymity*, the measured volume slightly decreases, but in almost all cases, the drop is limited to $10-15\%$ even with high values of z (yellow and cyan bar). In other words, *z-anonymity* would introduce a small measurement error in terms of traffic volume for these services, e.g., because less z users have requested their FQDNs in some ΔT (1 hour).

For a few cases, the difference is more pronounced; see, for example, *apple-dns.com*. In these cases, a single service/second-level domain holds a very large number of FQDNs, making them more likely to be anonymized. Indeed, in the case of *apple-dns.com*, I observe 1044 sub-domains, most of them differing uniquely for a two-digit code. Not shown in the figure, I observe a similar phenomenon for the CloudFront CDN, for which I observe 3115 sub-domains, each referring to a hosted website. In these cases, the measurement error introduced by z-anonymized increases. The issue can be solved by running *z-anonymity* on the second-level domains directly – i.e., configuring α-MON to release the second-level domain if it passes the *z-anonymity* checks. With this setup the measured volume is practically equal to the original traces, as I show in Figure 2.6b. Here, the users' privacy is still preserved, as α-MON hides the sub-domains, releasing only the z-private second-level domains. Yet, the value of the anonymized traces is increased substantially.



(a) *z-anonymity* on FQDN

(b) *z-anonymity* on second-level domain

Fig. 2.7 Per-service flows measured on the original and on a ($z = 10$)-anonymized trace.

I complete the above analysis with Figure 2.7, in which I broaden the bounds of pictures presented before. In the figure, I show with the red solid line the traffic volume (in terms of flows) for *all* services in the trace exceeding a minimum threshold of 1 000 flows (more than 1 000 names). They are sorted by popularity. The blue dashed line represents the volume as measured after *z-anonymity* with $z = 10$ and $\Delta T = 1$ h. I am interested in the deviation between the two lines, representing the measurement error. Considering *z-anonymity* on FQDNs (Figure 2.7), the error is minimal for the top-ranked domains, as already shown by the blue bars in Figure 2.6a. The deviation is still limited for the services with sizeable traffic, never exceeding an order of magnitude for those with at least (around) 10 k flows – top-200 names, left part of the figure. Clearly, the less frequent services are, the higher the chances α-MON anonymizes their FQDNs, thus increasing the measurement error. If *z-anonymity* is applied on second-level domains directly, Figure 2.6b shows that more reliable measurements are obtained also for less popular services. Also in this case, the error becomes high for infrequent services or those accessed by a very small number of users (see right side of the figure).

In summary, volumetric statistics are still reliable when targeting heavy-hitter services. For less frequent services, α-MON introduces a larger measurement error, as enforcing *z-anonymity* may lead to most occurrences of the associated FQDNs (or second-level names) to be obfuscated. By tuning z and ΔT, one can regulate the trade-off between privacy and data utility.

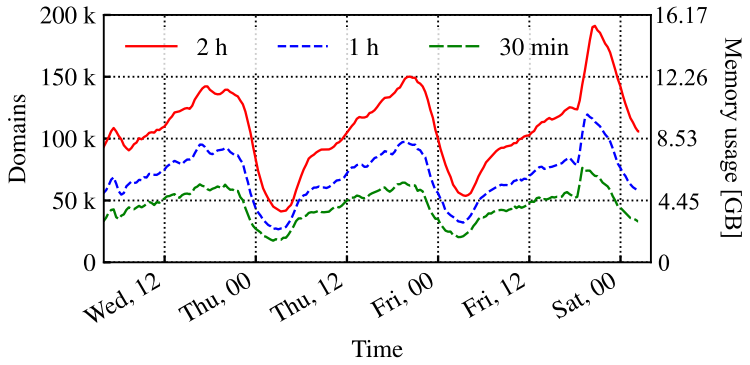


Fig. 2.8 Domains known by *z-anonymity* with different ΔT, in relation with the memory needed by α-MON.

2.5.3 Load on α-MON data structures

An important question for practically implementing *z-anonymity* is the number quasi-identifiers that *z-anonymity* has to track over time. This is fundamental to quantify its memory footprint and correctly size α-MON internal hash tables (see Section 2.4.4) as well as the ΔT parameter. For each quasi-identifier, indeed, I need to track the set of users associated in the last ΔT window.

Taking again FQDNs as an example, I consider the size of the set that *z-anonymity* must track – i.e., those FQDNs observed in the last ΔT interval. Figure 2.8 depicts results over time for our trace, considering three possible values for ΔT. After a short warm-up phase (not visible at this scale), the curves follow the daily trend of network usage. I observe a peak during evenings, when $\approx 150\,000$ unique FQDNs are seen in a two-hour interval – solid red line, see leftmost y-axis. No more than 100 000 (60 000) FQDNs appear with a ΔT of 1 hour (30 minutes). During the night, when traffic reduces, the number of active FQDNs is more than halved. I observe a sudden peak on the evening of the third day (a Friday) with almost 200 000 unique domains accessed in two hours. The rightmost y-axis of Figure 2.8 reports the memory footprint of the $QuasiID$ metadata, considering their actual size in our implementation. The memory usage is always below 17 GB for this setup.

Recall that the experiments refer to a traffic trace from a population of 8 000 users. However, given the nature of Internet traffic, where most flows are directed to few services, the set of domain names scales sub-linearly with the

Table 2.2 Packet traces.

	Flows (M)		Flows per-class (%)				Pktsize
Trace	TCP	UDP	HTTP	HTTPS	P2P	oth	avg
ISP-FULL	3.08	7.76	10.8	8.2	46.2	34.7	716
ISP-HDR	3.08	7.76	-	-	-	-	54
DNS	-	14.07	-	-	-	100	172

number of users. For example, during the peak hour, 1000 (or 3000) randomly selected users already contact 35000 (or 60000) FQDNs, while all 8000 users contact 150000 domains.

2.6 Performance Evaluation

I now evaluate the performance of α-MON in processing high speed traffic. I aim at evaluating how α-MON performance scales with the number of cores and the impact of different conditions, workloads and system parameters. I follow the standard benchmarking procedures defined in [49] for throughput tests.

2.6.1 Testbed and dataset

I instrument a testbed composed of a Traffic Generator (TG) and a Device Under Test (DUT). TG and DUT are each equipped with two quad-port Intel X710 10 Gbit/s network cards. TG replays traffic traces stored in `pcap` format, sending packets to DUT over a first set of 10 Gbit/s links. The DUT runs α-MON to anonymize network traffic that is sent back to the TG over a second set of 10 Gbit/s links.

DUT is a high-end server equipped with 4 Intel Xeon Gold 6140M processors and 512 GB of memory. The total number of physical cores is 72. I disabled hyperthreading to isolate α-MON performance when varying the number of cores.

The TG is a medium-sized server with no particular requirement except for a large amount of memory. Indeed, it is not trivial to read and send stored traffic traces at tens of Gbit/s with commercial solid-state drives whose read

speed is in the order of 4-5 Gbit/s. As such, I equipped the TG with 1 TB of RAM so that it can fit large traces in memory. I use DPDK-Replay to replay the traces on the selected network interfaces at the desired rate.[5] DPDK-Replay can loop over traces in memory, eventually replacing IP addresses on each pass, so to allow arbitrary benchmark duration.

I perform experiments using real traffic traces collected from an operational network (see Table 2.2). Packets are captured by instrumenting a Point-of-Presence of a European Internet Service Provider (ISP) that aggregates the traffic of about 8 000 households. I capture raw packets using a passive probe equipped with several high-end SSD disks.

For the first benchmarks, I use a 1-hour long trace captured at peak time. I obtain a 575 GB of packets that I call ISP-FULL. It contains 3.1 M TCP and 7.7 M UDP flows, with an average packet size of 716 B, for more than 800 M packets. This trace represents the typical workload that α-MON would face in an ISP network.

I process this trace to keep only up to TCP/UDP headers, removing payloads. This step results in a second packet trace – called ISP-HDR– in which packets are truncated to 54 B on average. I use this trace to benchmark the per-packet capture, processing and transmission speed of α-MON in a pessimist scenario composed of lots of small packets.

At last, I collect DNS traffic from the same network for one day. I use this trace to benchmark the *z-anonymity* module since each packet likely contains a $QuasiID$, e.g., a FQDN. This trace is 7.23 GB large, with more than 1 M packets. I call it DNS trace.

For each experiment, I seek the *throughput* [49], i.e., the fastest rate at which the count of frames transmitted by the DUT is equal to the number of frames sent by the TG. I progressively increase the TG sending rate using a binary search process. As I increase speeds, benchmarks require the TG to perform multiple passes on the original traces. All experiments are performed with $z = 10$ and $\Delta T = 60s$. Each benchmark lasts 3 minutes. I set the hash table *Hash(QuasiID)* size to 100 000 entries to maintain collision lists reasonably short (cfr. Figure 2.8).

[5] https://github.com/marty90/DPDK-Replay

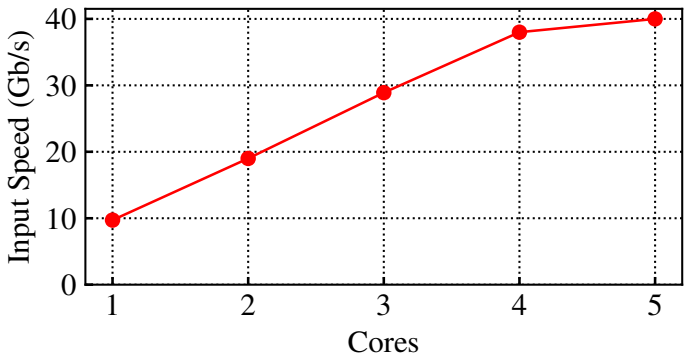
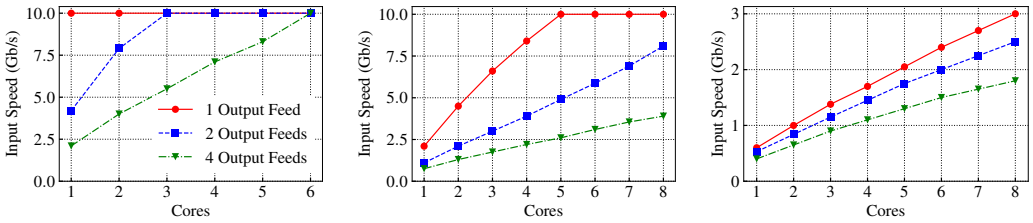


Fig. 2.9 Performance with four input and four output interfaces using the ISP-FULL trace.

2.6.2 Horizontal scalability

I first focus on α-MON horizontal scalability to understand how its throughput increases with the number of cores. Recall that each core manages an α-MON thread via DPDK. TG sends traffic to DUT using four 10 Gbit/s links. The DUT must anonymize packets before forwarding them on four output links. For each input interface, I configure one output feed on a dedicated output interface, thus, avoiding duplicating packets. Thanks to RSS load balancing, each of the N cores processes an average $1/N$ of the traffic from each input interface – $4/N$ in total, given I use 4 input interfaces. This load-balancing scheme makes the throughput to depend only on the aggregate incoming rate, regardless of the rates of single input interfaces. I employ the ISP-FULL trace for this experiment.

I report results in Figure 2.9, which shows the throughput versus numbers of cores. When α-MON runs on a single core, it handles around 10 Gbit/s. In my experiments, the throughput is equivalent if packets come from a single input link at line rate or spread on the four interfaces. With two cores, α-MON sustains 18 Gbit/s, and the performance scales linearly with additional cores, reaching 38 Gbit/s on four cores. With just five cores α-MON fully sustains 40 Gbit/s – i.e., all input interfaces at line rate. Unfortunately, my testbed does not allow higher rates due to the limited number of network interfaces, but I expect the performance to further increase before hitting the PCI bus bandwidth limit [50].



(a) ISP-FULL (b) ISP-HDR (c) DNS

Fig. 2.10 Performance with different traces and consumer numbers.

In summary, α-MON sustains $\approx 10\,\text{Gbit/s}$ per-core on a realistic traffic trace. Its performance scales to up 40 Gbit/s when using just five CPU cores, reaching line rate on all four input links.

2.6.3 Benchmark with other workloads

I evaluate α-MON performance under different workloads. I vary both the input traffic mixture and the number of consumers. In these experiments, the TG sends packets to the DUT using a single 10 Gbit/s link. I configure α-MON with one, two, or four output feeds, each of them anonymized using all available modules, but with different encryption keys. As such, α-MON not only has to make packet copies, but also performs all anonymization steps multiple times.

Recall that different traffic classes trigger different α-MON modules, resulting in performance variations. While the ISP-FULL trace is a *typical* workload that α-MON could face at an edge network, DNS represents an extreme scenario in which every packet triggers the *z-anonymity* module for FQDN. ISP-HDR is a second extreme scenario since all packets are small. It should not be observed in practice except for anomalous situations, e.g., during cyber attacks. ISP-HDR stresses α-MON packet replication capability toward multiple consumers as well as L2-L4 anonymization modules.

I show results in Figure 2.10. I report throughput for different traffic traces in separate figures, where lines indicate the number of output feeds. X-axes show the number of cores.

Figure 2.10a depicts the performance with the ISP-FULL trace. As already shown previously, a single core sustains 10 Gbit/s with a single consumer (solid

red line). The performance is reduced when α-MON has to feed multiple consumers. For a single CPU core (leftmost points), the throughput is reduced to 4 Gbit/s with two consumers (dashed blue line) and 2.4 Gbit/s with four consumers (dashed green line). The extra load imposed by the need for duplicating packets causes this degradation: DPDK allows zero-copy processing only when single output is required. Here, α-MON needs 3 cores to feed 2 consumers with 10 Gbit/s each, and 6 cores to feed 4 consumers. Note also how the throughput scales linearly with the number of cores in all cases. Here, contention on the *Hash(QuasiID)* has little impact.

Next, I use the ISP-HDR trace to stress α-MON packet copying, processing and forwarding. Whereas the TG sends out 1.7 million packets per second (Mpps) when replaying the ISP-FULL trace at 10 Gbit/s, ISP-HDR results in 23 Mpps. α-MON throughput naturally decreases. A single core handles no more than 2 Gbit/s in this scenario (Figure 2.10b - red curve). However, thanks to the scalable architecture based on RSS, α-MON throughput increases linearly with the number of cores – and 5 cores handle 10 Gbit/s when outputting traffic to a single consumer (red line). Similar to the previous scenario, the throughput is reduced when having multiple consumers (blue and green lines). A single core can sustain 1 (0.7) Gbit/s of the ISP-HDR trace with 2 (4) consumers. Yet, throughput continues to grow linearly with the number of cores. As such, a proper resource provisioning would allow α-MON to perform its tasks without loss also in these scenarios.

Next, I consider the DNS trace to stress the *z-anonymity* module. In Figure 2.10c I see that throughput further decreases. Remind that packets undergoing *z-anonymity* generate updates on various data structures to track the set of users associated with each quasi-identifier. Moreover, parsing the payload to recover quasi-identifiers is time consuming too (e.g., to extract FQDNs in DNS payloads). Figure 2.10c shows that a single core sustains 0.6 Gbit/s with one output feed. Again, the throughput increases almost linearly with the number of cores, and eight cores can handle 3 Gbit/s of DNS traffic. Here too, α-MON incurs a penalty for the packet copying in case of multiple consumers. The slightly sublinear scalability is due to the Mutex on the *Hash(QuasiID)* which slows down processing when a large number of cores are used.

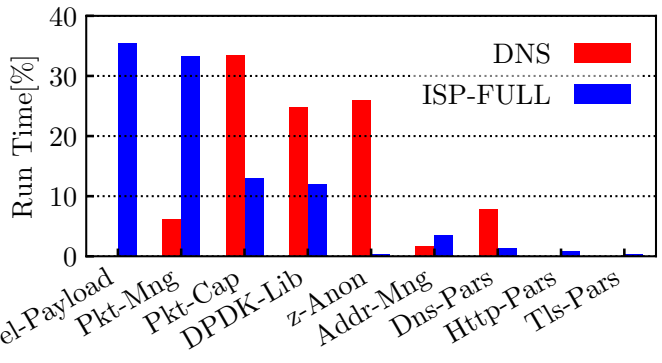


Fig. 2.11 Percentage of time spent on the most impacting modules.

In summary, α-MON can process 10 Gbit/s of typical ISP traffic with one core. Additional output feeds bring extra costs due to packet copying. A handful of cores allows achieving line rate in different scenarios. Worst-case scenarios, such as pure DNS traffic and millions of packets without payload, require a proper dimensioning of the system. α-MON scales linearly with the number of cores in all scenarios.

2.6.4 α-MON sub-module performance

I next show the impact on performance of α-MON components, by dissecting their execution time under different workloads. To this end, I instrument each α-MON sub-module with counters that use the CPU Time Stamp Counter to measure the elapsed time with a negligible performance penalty.[6] I then make experiments with the ISP-FULL and DNS traces, configuring α-MON with a single output feed, a single core, and replaying traffic at the sustainable rate – i.e., 10 Gbit/s for ISP-FULL and 0.6 Gbit/s for DNS.

Figure 2.11 shows the percentage of time spent on the main α-MON modules. I first notice how the nature of the traces determines different execution patterns. With ISP-FULL (blue bars), removing the payload from packets of insecure protocols (e.g., HTTP) absorbs most of the time due to a large number of memory write operations. Differently, with DNS (red bars), the *z-anonymity* module is invoked at each packet and accounts for 28% of the total execution

[6]The CPU Time Stamp Counter is a CPU register, thus, very fast to read.

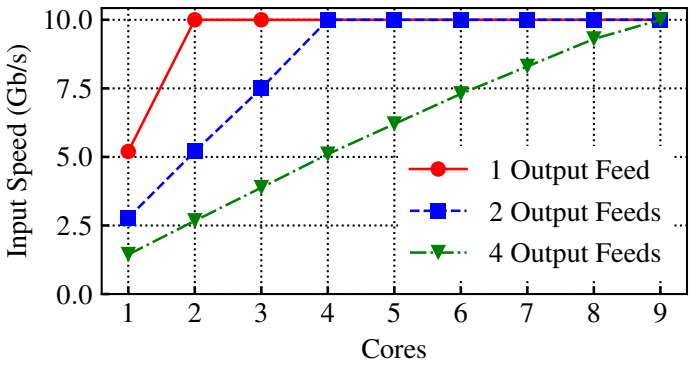


Fig. 2.12 Performance when applying *z-anonymity* on IP addresses and FQDNs (ISP-FULL trace).

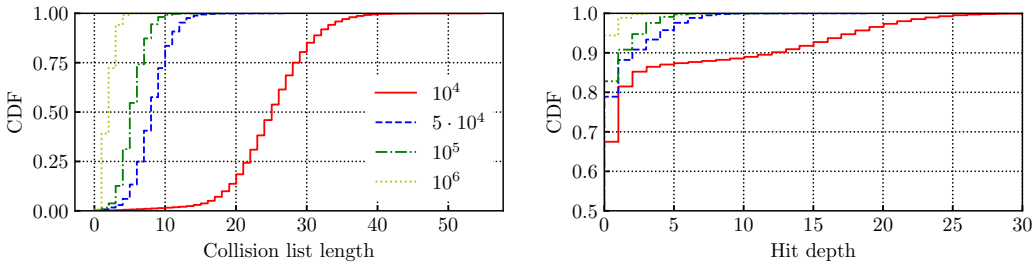
time (it is less than 1% for ISP-FULL). The packet capture (with DPDK) and general management routines in both cases have a significant impact, larger for DNS due to the smaller size of packets (see Table 2.2) and, thus, higher packet rate. Finally, note how IP address anonymization with CryptoPAN and protocol header parsing always have a marginal impact.

2.6.5 *z-anonymity* on other protocol fields

I now evaluate α-MON performance when applying *z-anonymity* on a wider range of protocol fields. Indeed, as described in Section 2.4, the *z-anonymity* module is flexible and can operate on different protocol fields, from FQDNs contained in DNS, TLS and HTTP, to the IP addresses of the contacted servers. In this experiment, differently from the previous cases, I make use of this feature and configure α-MON to apply *z-anonymity* on both FQDNs and IP addresses.[7] This imposes a high load on data structures. Indeed, the *z-anonymity* hash table is loaded with additional $QuasiIDs$ and the flow hash table needs to be used to make consistent decisions on a per-flow basis.

α-MON performance slightly decreases, as I show in Figure 2.12, in which I report the sustainable rate with different numbers of output feeds. α-MON achieves line rate with 2, 4 and 9 cores with 1, 2 and 4 output feeds, respectively.

[7] α-MON applies *z-anonymity* on server IP addresses only, obfuscating internal client addresses with CryptoPAN.



(a) Length of collision list for each QuasiID (b) Depth of hits in the collision list

Fig. 2.13 Analysis of the behaviour of Hash(QuasiID) collision list. Lines represent different hash table sizes.

When *z-anonymity* runs on FQDNs only (see Figure 2.10a), only 1, 3 and 6 are needed. In short, adding an additional field to *z-anonymity* decreases performance. As IP addresses are present in *every* single packet, decisions must be taken much more often than for FQDNs only. *z-anonymity* on IP address entails a $\approx 30\%$ performance drop due to the higher number of operations.

2.6.6 Tuning of the *z-anonymity* data structure

Here I study the impact of the data structure size for the *z-anonymity* module. Indeed, my implementation builds on the (large) hash table *Hash(QuasiID)* used to accommodate the quasi-identifiers, and, for each quasi-identifier, the ordered list of associated users in the last ΔT. Collisions on the hash table are handled with lists, whose length should be kept as short as possible to avoid performance impairments. This section evaluates the impact of the hash table size on the list length and the time *z-anonymity* spends iterating on them. To this end, I run multiple experiments using the DNS trace and varying the hash table size. During the experiments, I record, for each access to the *z-anonymity* data structure, the length of the collision list (if any) and the position at which α-MON found the matching quasi-identifier. The latter metric is particularly important given that α-MON handles collision lists in an LRU. As such, it is likely to find popular quasi-identifiers on the top of the list, avoiding exhaustive scans.

Figure 2.13a reports the distribution of the collision list length with different hash table sizes, from 10 k to 1 M elements. Clearly, a hash size much smaller

than the number of quasi-identifiers leads to long collision lists. When the size is 10 k (red solid line), lists are 25-element long in median, but can reach up to 40 elements. Large hash sizes reduce the length of collision lists, and I notice that quasi-identifiers are uniformly distributed among all hash buckets (not shown in the figure). However, collisions happen by chance, and, even with a 10 M elements (yellow dashed line), I sporadically find a handful of collisions.

Fortunately, α-MON does not need to fully scan collision lists, as a searched quasi-identifier is usually much before the tail of the list. Only for still unknown items (a mismatch), the list must be scanned exhaustively to ensure the quasi-identifier is not already present. In Figure 2.13b, I report the distribution of the position in the list of the matching element for each access to the data structure. Comparing it with Figure 2.13a, I notice how in most cases α-MON does not scan the lists entirely. Even with a small 10 k hash size (red solid line), on 80% of cases the matching item is found on the first or second position, and in less than 5% the search goes further than the 20^{th} position. With large sizes, the probability of evaluating more than 10 list elements becomes negligible.

In summary, the hash table must be sized to the deployment scenario to prevent collision lists from growing excessively. If properly done, *Hash(QuasiID)* allows *O(1)* access, with reasonably short collision lists, thanks also to the LRU policy which saves exhaustive scans.

2.7 Discussion on the z-anonymity approach

z-anonymity represents a new proposal for anonymizing sensitive information in network traffic. It shares with *k-anonymity*, *l-diversity* and *t-closeness* the idea that quasi-identifiers must be somehow controlled to prevent users' re-identification. No scheme can provide a guarantee of anonymity, and all schemes trade privacy with utility [51]. Indeed, publishing any data results in a potential privacy loss for individuals, and any anonymization technique makes data imprecise causing losses in potential utility. At last, efficient algorithms that provide anonymized data with such properties [52] are not well-fit for real-time and online usage as they make decisions based on the *global distribution* of quasi-identifiers. Like traditional approaches, *z-anonymity* provides a trade-off

and not full privacy guarantees. It however allows tuning the desired trade-off between privacy and data utility.

With *z-anonymity*, I propose a novel anonymization property that can be achieved in real-time and in an online fashion. As such, it is well-suited for network traffic anonymization. *k-anonymity* and similar approaches work on tabular data where the entire database (or a batch of data) are readily available. I instead want to anonymize a continuous stream of data and output the results in real-time. Notice that this differs from *k-anonymity* over data streams [53] – i.e., a system capable of applying *k-anonymity* on a stream database, where windows of data are considered. Other proposals [43, 44] work similarly, buffering data and releasing anonymized batches. Such approaches do not apply to my context since I cannot buffer lots of data while performing high-speed measurements in the network. Thus, I need to decide on a per-datum basis. Every decision has to be made in an atomic fashion, and the processed datum must be immediately available for later processing.

z-anonymity does not require to buffer data and scales very efficiently. As such, it is suitable for real-time deployments. To achieve that, *z-anonymity* is applied to each quasi-identifier in isolation as a performance trade-off. If the combination of multiple quasi-identifiers could lead to user re-identification, α-MON must be explicitly set to apply *z-anonymity* to the field combination. In fact, α-MON does not automatically search for such field combinations to increase performance.

Finally, in *z-anonymity*, the first $z-1$ user appearing in a ΔT would have their quasi-identifier values removed, while the z-th user would be the first one to have it visible. Nevertheless, she belongs to a set of at least z users, whose $z-1$ are unknown. In this sense, *z-anonymity* reduces the visibility of quasi-identifiers in the output stream.

2.8 Conclusion

In this chapter, I presented α-MON, a flexible and modular tool to anonymize network traffic according to a rich set of policies. I designed α-MON to be flexible and provide anonymized traffic to multiple legacy monitors with

different traffic visibility requirements, from security monitors to simple passive meters. A key innovation in α-MON is the implementation of *z-anonymity*, a stream-based traffic anonymization technique that obfuscates protocol fields that can be uniquely traced back to a small set of users. α-MON can search for them, for example, in the FQDNs present in DNS, TLS and HTTP traffic.

I designed a scalable architecture and efficient data structures to implement *z-anonymity* at line-rate speed on multiple 10 Gbit/s links. α-MON reaches high throughput in typical scenarios with few CPU cores. Even in worst-case scenarios α-MON scales linearly with the number of cores, thanks to its design based on DPDK. I quantified the impact of *z-anonymity* on common traffic measurements, showing that it introduces negligible measurement errors. For example, if applied before accounting traffic of websites, only for very infrequent sites the measured values would substantially differ from correct values due to the anonymization.

α-MON is available to the community as open-source software. As privacy and privacy-preserving analytics are gaining momentum, I believe α-MON can help researchers, network administrators and practitioners maintain visibility on network traffic while preserving users' privacy at the same time. Future work includes the development of mechanisms to find identifiers and quasi-identifiers in network traffic automatically as well as the analysis of the impact of *z-anonymity* on the operations of different classes of legacy monitors, which is addressed in the next chapter (Ch. 3).

Chapter 3

Generalizing z-anonymity as Zero-Delay Anonymization for Data Streams

In this chapter I will discuss the theoretical foundations on which *α-MON* is based, namely the *z-anonymity*, by presenting *z-anonymity: Zero-Delay Anonymization for Data Streams*, published in *2020 IEEE International Conference on Big Data* [54].

3.1 Introduction

Big data have opened new opportunities to collect, store, process and, most of all, monetize data. This has created tension with privacy, especially when it comes to information about individuals. We live in the data era, where a big part of our life is readily available in digital format, from our online activity to our location history, from what we buy to how we spend out free time [55]. Recently, legislators have introduced privacy laws to regulate the data collection and market, with notable examples of the General Data Protection Regulation (GDPR) in EU, or the California Consumer Privacy Act (CCPA) in the US.

The classical approach to publish personal information is anonymization, i.e., generalizing or removing data of the most sensitive fields. Thanks to this,

Privacy-Preserving Data Publishing (PPDP) has gained attention in the last decade [56]. It is now even more popular (and critical) with the birth of data markets where data buyers can have access to large collections of data about individuals. Removing the user's *identifiers* (name, social security number, phone number, etc.) is not sufficient to make a dataset anonymous. Indeed, an attacker can link a user's apparently harmless attributes (such as gender, zip code, date of birth, etc.) called *quasi-identifiers* (QIs) to a (possibly even public) background knowledge. In this way, the attacker can re-identify the person and gain access to further sensible information from the dataset (disease, income, etc.) called *sensitive attributes* (SAs) [57]. Famous is the de-anonymization of Netflix public dataset [58] based on the study of QIs.

Researchers proposed several properties that anonymized data should respect to avoid re-identification, the most popular of which is the k-anonymity [23]. Despite its limits, it remains the golden standard for anonymization. k-anonymity imposes that the information of each person contained in the release cannot be distinguished from at least $k-1$ individuals whose information also appears in the release. k-anonymity is conceived for tabular and static data. In other words, the dataset must be completely available at anonymization-time. Extensions to a streaming scenario have been proposed, where continuously incoming records are processed, typically using sliding windows [44]. In this case, the new records are temporarily stored, processed and released after an unavoidable delay. However, for specific applications it is fundamental to avoid any processing delay. For example, for network traffic, where it is unfeasible to store packets for a long time, or location history, if a real-time (but anonymous) stream shall be used for, e.g., mobility optimization.

3.1.1 My Contribution

This chapter explores in a more theoretical and rigorous way the *z-anonymity*. As already introduced in 2, it is designed to work with data streams and can be achieved with zero-delay (hence the choice of the letter z instead of k). I assume to observe a raw stream of data, in which users' new attributes are published in real-time as they are generated. Apart from network traffic (which has been already discussed), I can assume, for instance, a new transaction in their credit card, a new position of their car, or a new website

they visit. These attributes are QIs, and, when accumulated over time, may allow users' re-identification.

For the sake of simplicity for the reader, I report here some concepts that have already been introduced in the previous chapter. *z-anonymity* builds on the same idea of k-anonymity. When a new attribute arrives, it is released only if at least $z-1$ individuals have presented the same attribute in the past window Δt. Otherwise, it is blurred. *z-anonymity* is weaker than k-anonymity since it cannot guarantee that at least $k-1$ users present the same *combinations* of QIs (i.e., the aggregated record). Implementing *z-anonymity* in real-time at high speed requires ingenuity, especially considering the large number of attributes the system deals with - i.e., the high-dimensional data problem, which is one of the problems hampering k-anonymity too [59]. In this chapter, I show that *z-anonymity* can be obtained both with zero-delay and in an efficient way when employing a scalable implementation and appropriate data structures. Lastly, I present a probabilistic framework to map z- into k-anonymity properties. I find out that *z-anonymity* can provide k-anonymity with desired probability, for appropriate values of z.

There are various examples of application of *z-anonymity*. For instance, I originally proposed it for internet traffic analysis, as discussed in Ch. 2, or similarly, the user browsing history, the credit card history, and the location history that offer rich information for companies which want to access as quickly as possible, i.e., datum after datum, without waiting for records to be aggregated. For instance, recent credit card transactions can be useful for fraud detection or shopping recommendations; the browsing history for personalized advertisements or market intelligence; the location history to promptly optimize the mobility, or study patterns of real-time traffic.

In the remainder of the chapter, I formalize the *z-anonymity* property and present an approach to implement it efficiently and in real-time (Section 3.2). I then propose a probabilistic model to derive k-anonymity properties from z-anonymized streams (Section 3.3), and study the effect of the different parameters (Section 3.4). I then apply the model to the browsing history use case (Section 3.5). Finally, I discuss the limitations of my approach and future work (Section 3.6) and draw the conclusions (Section 3.7).

3.2 z-anonymity

3.2.1 The z-anonymity approach

I work on a data stream, in which I continuously receive observations that associate users with a value of an attribute. I define an observation as (t, u, a), which indicates that, at time t, the user u exposes an attribute-value pair a.[1] For example, if *Sex* is the attribute, and *Female* is the value assumed by the attribute of user u at time t, then a is the pair *(Sex, Female)*. Attributes can be related to whatever field: a visit to a web page, a GPS location, a purchase, etc. I consider attributes a as *quasi-identifiers*, while *sensitive-attributes* are not present. I want to keep private those values of attributes associated with a small group of users. As mentioned in Definition 1, an attribute-value pair a is z-private at time t if it is associated with less than z users in the past Δt time interval.

Notice that the same attribute a can be both z-private and not z-private at different time t.

If the anonymized dataset hides all z-private attribute-value pairs, it achieves *z-anonymity*. Recalling Definition 2, a stream of observations is z-anonymized if all z-private attribute-value pairs are obfuscated, given z and Δt.

In other words, the attributes that are associated with less than z users in the past Δt shall be obfuscated, i.e., removed or replaced with an empty identifier. The goal is to prevent rare values of attributes to be published, thus reducing the possibilities of an attacker to re-identify a user through unusual attributes.

z and Δt are system parameters that can be tuned to regulate the trade-off between data utility and privacy. This allows *z-anonymity* to adapt to the needs of the desired use case, resulting in a flexible paradigm that can be used in many different fields. A large z and a small Δt result in the majority of attributes to be anonymized, while a small z or a large Δt allows rare values to be possibly released. Δt regulates the memory of the system.

[1] Here I will use attribute and attribute-value pair interchangeably.

It is important to recall that *z-anonymity* acts in an attribute-by-attribute fashion, not considering their combinations as in the k-anonymity property. Hence, it is interesting to study which guarantees the *z-anonymity* algorithm offers in a global perspective, i.e., which assumptions it is possible to make on the overall privacy properties (e.g., it terms of k-anonymity) of the output.

3.2.2 Implementation and complexity

The *z-anonymity* property can be achieved in real-time with zero delay using a simple algorithm based on efficient data structures. I propose to generalize the approach presented in Ch. 2: the attribute-value pairs a are stored as a hash table $\mathcal{H}$, with linked lists to manage collisions. Each value $\mathcal{H}(a)$ in the hash table contains three elements:

- metadata about a;
- a Least Recently Used list LRU_a of tuples (t, u);
- a hash table $\mathcal{V}_a$ for the users.

The idea is to minimize the time spent searching into the data structures, therefore reducing the memory accesses. By assuming that the number of attributes a has the same order of magnitude of the hash structure dimension, collisions are infrequent, and consequently, the total computational cost is $O(1)$ for each incoming observation.

The $\mathcal{H}(a)$'s metadata include the counter c_a and the reference for the LRU_a first and last attribute. Referring to Algorithm 1, once an observation (t, u, a) arrives, the value a should be inserted in the hash table, if not already present (lines 2-6), otherwise an update should be performed (lines 7-21). The hash value is calculated and the access to the table is done in $O(1)$.

If the user u comes with attribute a for the first time in the previous Δt, the user u is inserted into $\mathcal{V}_a$ in $O(1)$, c_a is increased by one and the tuple (t, u) is inserted on top of the LRU_a in $O(1)$ thanks to the aforementioned references (lines 8-11). If u was instead already present in $\mathcal{V}_a$ and in LRU_a with value (t', u), I replace t' with t and the tuple (t, u) is moved on the top of the LRU_a. Again all is done in $O(1)$ (lines 12-14).

Algorithm 1 Pseudo code of the algorithm to implement *z-anonymity*.

1: **Input:** (t, u, a)
2: **if** $a \notin \mathcal{H}$ **then**
3: $\mathcal{H} \leftarrow \mathcal{H} \cup a$ *//new attribute: insert it for the first time*
4: $\mathcal{V}_a \leftarrow \{u\}$ *//insert new user u*
5: $LRU_a \leftarrow (t, u)$
6: $c_a = 1$
7: **else**
8: **if** $u \notin \mathcal{V}_a$ **then**
9: $\mathcal{V}_a \leftarrow \mathcal{V}_a \cup \{u\}$ *//insert new user u*
10: $c_a \leftarrow c_a + 1$ *//add new user*
11: $LRU_a \leftarrow (t, u)$
12: **else**
13: $(t', u) \leftarrow (t, u)$ *//update timestamp of user u*
14: move (t, u) on top of LRU_a
15: **end if**
16: **end if**
17: *//Always evict old users*
18: **for** $((t', u') = \text{last}(LRU_a);\ t' < t - \Delta t;\ (t', u')=\text{next})$ **do**
19: remove (t', u') from LRU_a
20: remove (u') from $\mathcal{V}_a$
21: $c_a \leftarrow c_a - 1$
22: **end for**
23: **if** $(c_a \geq z)$ **then**
24: OUTPUT (t, u, a)
25: **end if**

Last, to evict old entries and consequently decrease c_a, I traverse the LRU in reverse order: I remove each tuple (t', u') where $t' < t - \Delta t$, and I decrease c_a accordingly (lines 17-21). At last, if $c_a \geq z$ the observation (t, u, a) is released (lines 23-24).

k-anonymity has been proved [60] an *NP-Hard* problem. Differently, *z-anonymity* property can be achieved for each observation with $O(1)$ complexity with properly sized hash-tables.

3.3 Modelling *z*-anonymity and *k*-anonymity

I now study the relationship between the *z-anonymity* and k-anonymity properties. In particular, I quantify how a z-anonymized dataset could result

in a k-anonymity release with a certain probability. Intuitively, *z-anonymity* ensures that each published value of an attribute a is associated at least with z users in the past time interval, while, with k-anonymity, any given record (i.e., the combinations of all user's attributes) appears in the published data at least k times. Recall that with high-dimensional data, the set of attribute-value combinations becomes extremely high, thus making k-anonymity tricky to guarantee. Here I show that with a proper choice of z, it is possible to release data in which users are k-anonymized.

I define a simple model where users generate a stream of attributes. Each attribute has a given probability of appearance that reflects its different popularity. I assume few attributes are very popular, with a long tail of infrequent attributes that may seldom appear. This often happens in real-world systems that are governed by power-law distributions [61].

3.3.1 User and attribute popularity model

I consider a system in which a set of $\mathcal{U}$ users can access a catalog of $\mathcal{A}$ attributes. Let $U = |\mathcal{U}|$ and $A = |\mathcal{A}|$.

Users generate a stream of information, exposing in real-time the attribute they have just accessed. For instance, this reflects a location tracking system in which black boxes installed on a fleet of vehicles periodically exports each car location; or operating system telemetry that periodically reports which application is running; or network meters reporting which website a user is visiting. The system collects *reports* in the form of the tuple (t, u, a), i.e., at time t, the user $u \in \mathcal{U}$ exposes the attribute $a \in \mathcal{A}$. For simplicity, I assume that users are homogeneous and all reports are independent, so that the probability of getting a report, only depends on the value assumed by a.[2] In particular, I assume any user u exposes the attribute a with a given rate λ_a, with exponential inter-arrival time. Hence, given the time interval Δt, the number of times a user exposes an attribute a is modeled as a Poisson random variable R_a with parameter $\lambda_a \cdot \Delta t$ ($R_a \sim Poisson(\lambda_a \cdot \Delta t)$).

[2] I can relax this assumption, e.g., by considering classes of users. I leave this for future work.

I denote as X_a the random variable describing whether a user exposed at least once attribute a in a time interval Δt. X_a assumes value 1 if a user exposes a in Δt, 0 otherwise. I note that $X_a \sim Bernoulli(p_a^X)$, where p_a^X is the probability that a user exposes attribute a at least once in the past Δt. It is straightforward to compute p_a^X given λ_a and Δt as:

$$p_a^X = P[R_a \geq 1] = 1 - P[R_a = 0] = 1 - e^{-\lambda_a \cdot \Delta t} \tag{3.1}$$

3.3.2 Applying *z-anonymity*

I study how a stream of data modeled as above appears when released respecting *z-anonymity*. With *z-anonymity*, z-private attributes at time t are removed. Namely, if less than other $z-1$ users are associated with a in the previous Δt, the current association is blurred. I here define the event of a report (t,u,a) to be published when exposed as a random variable O_a. I have that O_a is a Bernoulli random variable with parameter p_a^O.

$$p_a^O = P[O_a = 1] = P\left[\sum_{v \in \mathcal{U} \setminus u} X_a \geq z - 1\right] \tag{3.2}$$

Given our assumption of independent and homogeneous users, I am summing $U-1$ times the same random variable X_a. I remove one user since I are checking the *z-anonymity* for the report (t,u,a). Hence one user is already involved by construction. Since X_a is a Bernoulli with success probability p_a^X, its sum results in a Binomial distribution, measuring the number of occurrences in a sequence of $U-1$ independent experiments $\sum_{v \in \mathcal{U} \setminus u} X_a \sim \mathcal{B}(U-1, p_a^X)$.

Starting from Equation (3.2) and using the probability mass function of the Binomial distribution I can derive p_a^O as:

$$p_a^O = 1 - \sum_{i=0}^{z-2} \binom{U-1}{i} \left(p_a^X\right)^i \left(1 - p_a^X\right)^{U-1-i} \tag{3.3}$$

Similar to what I did in Section 3.3.1, I denote as Y_a the random variable describing if a user published at least once attribute a in a time interval Δt. I note that $Y_a \sim Bernoulli(p_a^Y)$, where p_a^Y is simply:

$$p_a^Y = P[X_a = 1] \cdot P[O_a = 1] = p_a^X \cdot p_a^O$$

The set of the random variables describing the presence or absence for all the possible attribute-value pairs $a \in \mathcal{A}$ for a user is denoted as $\bar{Y} = \{Y_a\}_{a \in \mathcal{A}}$. Again this is equal for all users, being them homogeneous.

3.3.3 The attacker point of view

I assume an attacker observes the z-anonymized output streams for all users $u \in \mathcal{U}$ for a time $N\Delta t$ with $N \in \mathbb{R}^+$ (for simplicity, in our model I considered $N \in \mathbb{N}, N \geq 1$). Hence, in our scenario, the attacker can accumulate the output for a time span possibly much larger than the parameter Δt. Similarly to Y_a, I can thus define the random variable Y_a^N, that models whether a user exposed *and* published attribute a at least once during the total observation period $N\Delta t$. It is clear that Y_a and Y_a^N are strongly related. In fact I have $Y_a^N \sim Bernoulli(p_a^N)$, where the parameter p_a^N can be computed as follows:

$$p_a^N = [1 - (1 - p_a^Y)^N]$$

This is because for a user u to expose and publish an attribute a in the period $N\Delta t$, (s)he has to be associated with a value 1 of Y_a at least in one of the N periods Δt long. At the end of the period $N\Delta t$, the attacker has observed U users hence obtaining U realizations $\overline{y^N}$ of the random variable $\overline{Y^N} = \{Y_a^N\}_{a \in \mathcal{A}}$ including all the possible attributes.

The attacker will not know the random variable $\overline{Y^N}$, and will observe only realizations of it. Let us denote as y_a^N a realization of the random variable Y_a^N and as $\overline{y^N} = \{y_a^N\}_{a \in \mathcal{A}}$ a realization of the random variable $\overline{Y^N}$.

3.3.4 Getting to k-anonymity

I want to check to what extent a z-anonymized stream of a user satisfies also k-anonymity property in the whole stream of U users. Given a specific realization $\overline{y^N}$ of a user, our goal is to derive the probability to observe at

least other $k-1$ users in $\mathcal{U}$ having the same realization $\overline{y^N}$. If this happens, the system lets k users release the same attributes and thus they cannot be uniquely re-identified, resulting k-anonymized.

Let us consider first the probability that two realizations of Y_a^N are equal. Let us denote the two realizations, related to two users u and v, as $y_a^N(u)$ and $y_a^N(v)$. The probability is simply $(p_a^N)^2+(1-p_a^N)^2$ because either both take the values of 1, or both take the value of 0. Remind that the users are assumed to act independently. The probability that two users have the same realization of $\overline{Y^N}$ is then the following:

$$p^Q = P\left[\overline{y^N(u)} = \overline{y^N(v)}\right] = \prod_{a\in\mathcal{A}}\left((p_a^N)^2+\left(1-p_a^N\right)^2\right)$$

where $\overline{y^N(u)}$ and $\overline{y^N(v)}$ are the two realizations of $\overline{Y^N}$. The parameter p^Q can be seen as the parameter of a Bernoulli random variable Q describing whether two realizations are equal (assuming value 1) or not (assuming value 0).

Finally I define the probability that a given realization $\overline{y^N(u)}$ satisfies the k-anonymity property. Hence, it means that there are at least $k-1$ other users with the same realization. I denote this probability as p_{k-anon}.

$$p_{k-anon} = P\left[\sum_{v\in\mathcal{U}\setminus u} Q \geq k-1\right]$$

Then p_{k-anon} is the probability that at least other $k-1$ realizations are equal to the one studied. Again, as in Equation (3.2), $\sum_{v\in\mathcal{U}\setminus u} Q$ follows a Binomial distribution of $U-1$ experiments with probability p^Q. Then I can derive p_{k-anon} as in Equation (3.3):

$$p_{k-anon} = 1-\sum_{i=0}^{k-2}\binom{U-1}{i}\left(p^Q\right)^i\left(1-p^Q\right)^{U-1-i}$$

In summary, our model describes the probability that a data stream undergoing *z-anonymity* results in dataset which respects the k-anonymity property. Although I can only provide a probabilistic guarantee that the released data

Table 3.1 Terminology used to model *z-anonymity* and k-anonymity.

Term	**Definition**
$\mathcal{U}$,U	Set and number of users
$\mathcal{A}$,A	Set and number of attribute-value pairs
Δt	The time interval length used for evaluating *z-anonymity*
N	Length of the data stream, in multiples of Δt on which I test the k-anonymity
λ_a	Exposing rate for attribute a
R_a	Random variable counting number of times a user exposes attribute a in Δt. $R_a \sim Poisson(\lambda_a \cdot \Delta t)$
X_a	Random variable representing whether a user exposes attribute a in Δt. $X_a \sim Bernoulli(p_a^X)$
O_a	Random variable representing whether a report (t,u,a) is published when exposed. $O_a \sim Bernoulli(p_a^O)$
Y_a	Random variable representing whether a user published at least once attribute a in Δt. $Y_a \sim Bernoulli(p_a^Y)$
Y_a^N	Random variable representing whether a user published at least once attribute a in $N\Delta t$. $Y_a^N \sim Bernoulli(p_a^N)$
$\overline{Y^N}$	Set of random variables $\{Y_a^N\}_{a\in\mathcal{A}}$
Q	Random variable representing whether two realizations of $\overline{Y^N}$ are equal. $Q \sim Bernoulli(p^Q)$
p_{k-anon}	Probability that a realization of $\overline{Y^N}$ satisfies k-anonymity property

will be k-anonymized, I can study and control this probability as a function of the parameters.

3.4 Comparing *z*-anonymity and *k*-anonymity

In the following, I show the impact of the system parameters on the k-anonymity and *z-anonymity* properties. In my model, I assume a small set of popular attributes and a large tail of infrequent ones. This allows us to catch the nature of systems where users are more likely to expose top-ranked attributes, but there exist a large catalog. As such, I choose that the λ_a for all attributes follow a power law in function of their rank. Let us suppose attributes are sorted by rank, and the most popular attribute is a_1 and the least popular a_A. I impose $\lambda_{a_1} = 0.05$ and set the remaining λ_a as the power-law function $\lambda_{a_r} = {}^{0.05}/_{r}$, where r is the rank of attribute a_r. The p_a^X value is evaluated as described in Equation (3.1) - for the sake of simplicity, I consider $\Delta t = 1$ unit

Table 3.2 The default values used for the model.

Variable	Default Value
U	50 000
A	5 000
λ_{a_r}	0.05 / r
N	24
z	20
k	2

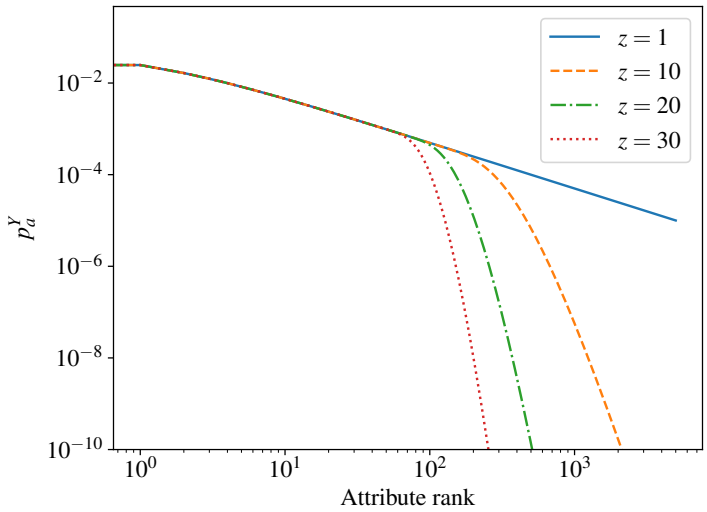
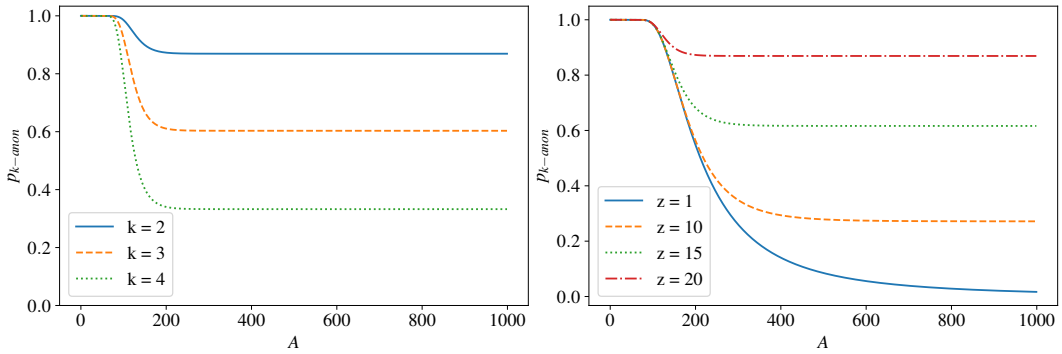


Fig. 3.1 The probability p_a^Y for a user to publish attribute a in Δt, according to its rank.

of time. Notice that the different attributes are independent and $p_{a_r}^X$ is not a distribution probability mass function, hence it does not have to sum to 1.

I have defined a model that describes the probability that in the released data, satisfying *z-anonymity*, a user has at least $k-1$ other users with the same set of associated attributes. Formally speaking, $p_{k-anon} = \mathcal{F}(U, A, \lambda, N, z, k) \rightarrow [0, 1]$. As such, $\mathcal{F}$ gives the probability a generic user is k-anonymized in the released data. Each of the above parameters has an impact on the output probability p_{k-anon}. Here, I study the impact of different combinations of parameters. Where not otherwise noted, the default parameters listed in Table 3.2 are used.



(a) p_{k-anon} changing k ($z = 20$) (b) p_{k-anon} changing z ($k = 2$)

Fig. 3.2 he impact of A on p_{k-anon}, considering both different k and z values.

3.4.1 The impact of the attribute rank

I first focus on the p_a^Y, i.e., the probability of observing at least once the attribute a in a Δt, for a given user, in the released data, after z-anonymization. Figure 3.1 shows the p_a^Y in function of the attribute rank. Remind that the popularity of attributes follows a power law, since $\lambda_{a_r} \approx r^{-1}$. Indeed, the blue solid line in the figure shows the probability of observing an attribute in case $z = 1$, i.e., no anonymization (equal to p_a^X). The curve appears as a straight line, representing a power law on the log-log plot. When enabling *z-anonymity* ($z > 1$), I notice that the probability of observing uncommon attributes abruptly decreases with an evident knee. For example, if I observe the curve for $z = 20$ (green dashed line in the figure), already the 300^{th}-ranked attribute is observed with a probability below 10^{-6}, while it appears on the original stream with 10^{-3}. A higher z moves the knee of the curve closer to the top-ranked attributes. In other words, the figure shows how *z-anonymity* operates in preventing uncommon attributes from being released. Indeed, those attributes are released only when enough users are exposing them, hence very rarely.

3.4.2 The impact of A

In Figure 3.2, I study the impact of the size of the catalog of attributes $\mathcal{A}$. In Figure 3.2a I show in a *z-anonymity* dataset how the probability p_{k-anon} of a user being k-anonymized varies with $\mathcal{A}$. To this end, I perform different

simulations with increasing numbers of attributes A. I consider a system where only the top A ranked attributes exist. Intuitively, with a large number of attributes, it is harder to find users with the same output attribute set $\overline{y^N}$. However, my assumption of a long tail of infrequent attributes plays with us. indeed, the probability of observing them rapidly goes to to 0 (see Figure 3.1), and, as such, these attributes rarely appear in the users' released sets. Figure 3.2a shows this behavior with $k = 2, 3, 4$, while keeping constant values of z and U. With a very small catalog of top-100 or less attributes, users are k-anonymized with reasonable certainty, being very likely to observe multiple users with the same set $\overline{y^N}$. When A increases, I start releasing less-popular attributes. The number of possible attribute combinations thus explodes exponentially[3], and *z-anonymity* starts showing its effects. Focusing, for example on the orange dashed curve for $k = 2$, when A exceeds 100, the probability of finding 1 or more identical users to a given one suddenly decreases. However, it settles to approximately 0.9 with $A > 100$, clearly showing the effect of *z-anonymity*. The infrequent attributes are not released, and, as such, this limits the explosion of the possible combinations. Further enlarging A does not affect p_{k-anon}, as the attributes in the tail are anyway not published. Increasing the value of k results in lower probability of satisfying k-anonymity property.

For comparison, in Figure 3.2b I report the effect of finding at least an identical user to a given one with different values of parameter z of *z-anonymity*. Similarly to the other cases, p_{k-anon} starts at 1, when few attributes are present, and the number of their possible combinations is low. When A increases, less frequent attributes start to appear. The possible combinations of attributes explode exponentially. With $z = 1$, i.e., no *z-anonymity* in place, the probability of finding identical users rapidly goes to 0. Enabling *z-anonymity*, I prevent rare attributes to be released, thus reducing the possible combinations. The higher z, the higher the p_{k-anon}.

In summary, *z-anonymity* allows k-anonymity to be satisfied with a non-zero probability, even with a long tail of attributes.

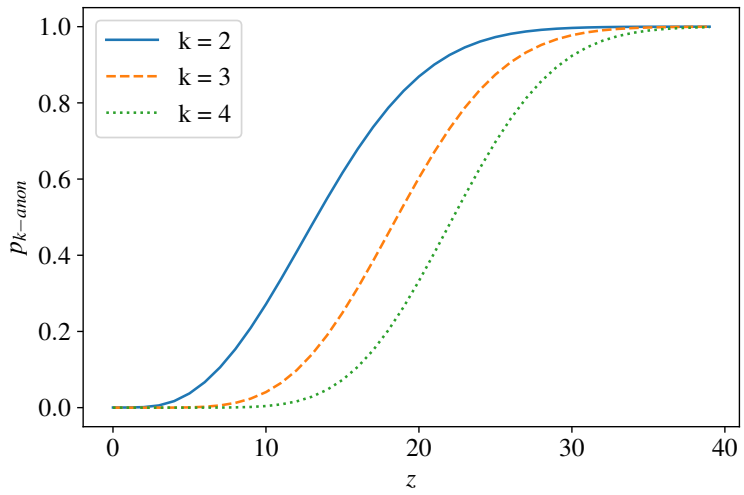


Fig. 3.3 The impact of z on p_{k-anon} for different k values.

3.4.3 The impact of z

I now evaluate the impact of z on the p_{k-anon}. In Figure 3.3, I report how different values of z result in different probabilities for a given user to be k-anonymized, i.e., there are at least $k-1$ other users with the same set of released attributes. The other parameters are fixed to the values shown in Table 3.2, and different lines correspond to different values of k. Intuitively, the larger is z, the higher is p_{k-anon}. Focusing on $k=2$ (blue solid line), p_{k-anon} increases starting from $z=4$. With $z=20$, the probability of finding at least a user with an identical set of released attributes is already 0.8. When $k>35$, p_{k-anon} approaches 1 for the three curves, giving the almost certainty that the whole release is k-anonymized (for $k=2,3,4$). In other words, it is possible to choose a proper z to enforce a desired k and p_{k-anon} on the released data.

3.4.4 The impact of U

Next, I study in Figure 3.4 how the number of users U impacts the privacy of the released data. If I only increase the number of users U, not shown in the Figure, there is a higher chance that some users have even rare attributes released, breaking thus k-anonymity. This would happen because a large number of users would cause even less-popular attributes to overcome the

[3]The attribute combinations increase as 2^A.

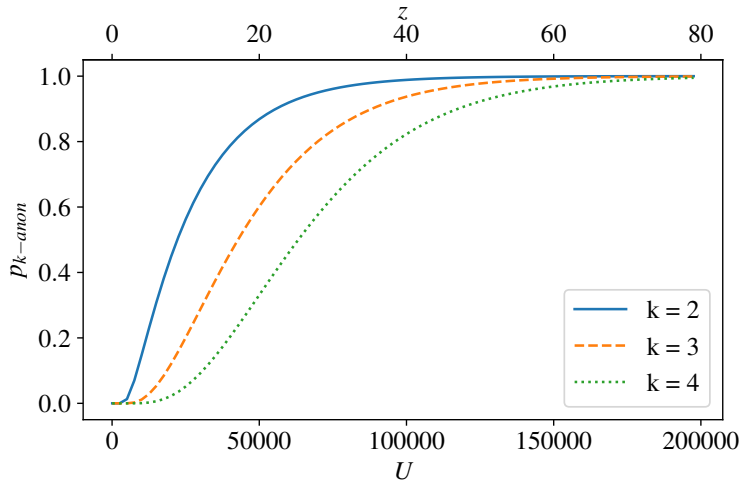
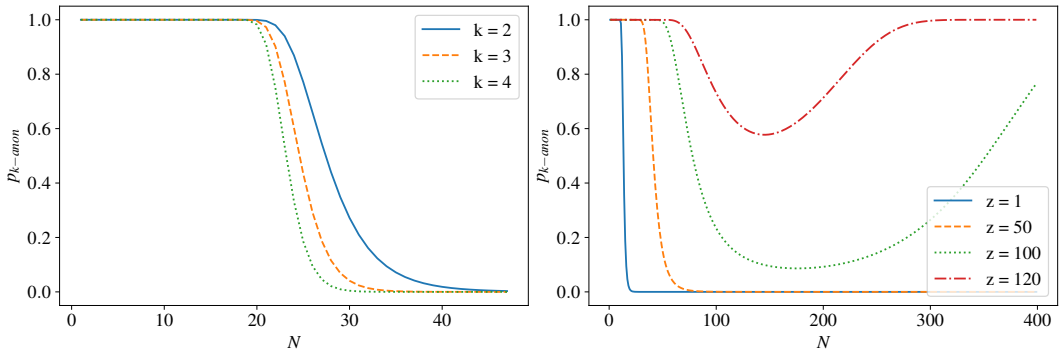


Fig. 3.4 The impact of U and z on p_{k-anon} for different k values ($z = 20$).

z threshold, increasing the number of possible combinations, and decreasing p_{k-anon}. Hence, for a fair comparison, z is set proportional with U, and I report it on the upper x-axis of Figure 3.4. Again, A is fixed to 5 000. Focusing on $k = 2$ (blue solid line), I notice how p_{k-anon} grows quickly with U. With $U = 22\,000$ (and $z = 9$), the probability of a user of having another user with identical attributes is already 0.5. p_{k-anon} keeps growing, even if at a lower pace, reaching value very close to 1 with $U = 100\,000$. This result shows that a large number of users leads to better guarantees of k-anonymity as far as z is set proportionally to U.

3.4.5 The impact of N

Finally, Figure 3.5 shows the impact of the observation time of the attacker (N), defined for simplicity in multiples of Δt. The figure quantifies how increasing N affects p_{k-anon}. In Figure 3.5a N varies on the x-axis, while different lines represent different k. Intuitively, having a larger observation time makes it more difficult for users to be k-anonymized, since the probability that rare attributes are released increases, and, thus, the number of attribute combinations. When an attacker can access enough z-anonymized data, p_{k-anon} drops. Looking at the blue solid line for $k = 2$, after $N = 22$ periods of Δt, the probability of finding identical users starts falling, reaching 0 with $N = 45$. I



(a) p_{k-anon} changing k ($z = 20$) (b) p_{k-anon} changing z ($k = 2$)

Fig. 3.5 The impact of observation time N on p_{k-anon}, considering both different k and z values.

observe a similar behavior with higher values of k (dashed lines), for which the decrease starts earlier and it is steeper.

Figure 3.5b shows different insights, observing the impact of the attacker obtaining data in a longer time window. Here, I fix $k = 2$, and I draw different lines for different z, with N up to 400. With $z = 1$, no k-anonymity can be guaranteed as soon as the attacker observes the data for $N > 3$. *z-anonymity* preserves k-anonymity for longer time (e.g., up to $N = 70$ for $z = 120$). This suggests to use z-anon in combination with other privacy preserving approaches, e.g., user ID rotation or randomization after $N\Delta t$ time. Interestingly, with larger values of z, p_{k-anon} grows again as the observation time increases. This happens because, sooner or later, the most popular attributes will be exposed *and* published by almost every user. Hence, the observations $\overline{y^N(u)}$ will be mostly composed of 1s, and thus most likely be equal to others. For this phenomenon to occur within a reasonable observation time, z must be large enough to just consider most popular attributes, that will take less time to be exposed by almost every user.

3.5 A practical use case: the visits to websites

Differently from Ch. 2, in this section I explore an other practical use case for the *z-anonymity*: the users' navigation data. To this end, I use the data gathered on a real network to set the parameters of my model. I build on passive

measurements collected by Tstat [62], a passive meter that collects rich flow-level records, including hundreds of statistics on the monitored traffic. Essential to my analysis, Tstat builds a log entry for each TCP connection observed on the network, and, for each, it reports, among other statistics, the IP address of the client, a timestamp and the domain name of the server as indicated on the HTTP or TLS headers.[4] I use the entries collected over one day in 2018 in a Point Of Presence of a European ISP aggregating the traffic of approximately 10 000 households. To filter those websites carrying very little information, such as content delivery networks, cloud providers or advertisement, I keep only those websites included in the top-1 Million rank by Alexa[5] and not belonging to the aforementioned categories. For privacy reasons, I encrypted the client identifiers, i.e., the IP addresses, with the Crypto-PAn [63] algorithm, rotating the encryption keys every day.

I use 1 day of collected data to estimate values of the parameters. I assume $\Delta t = 1\,hour$ and $N = 24$. I obtain $A = 27\,482$ and $U = 9\,670$, and I estimate directly the p_a^X for each attribute (a website in this case).[6] Then, I setup my analysis with these obtained parameters, running my probabilistic framework and showing the results I obtain.

In Figure 3.6, I show the probability p_a^Y of observing the attribute a, for a given user, in the released z-*anonymity* data. The solid blue line corresponds to $z = 1$, i.e., no anonymization, thus reporting the popularity of websites in the dataset. The most popular website is *google.com*, which has $p^X_{google.com} = 0.34$, meaning that in 1 hour any of the users will visit this website at least once with this probability. There are some very popular websites, with the top-7 ranked having $p_a^X > 0.1$. In the tail, I find 15 464 websites accessed by only one user on the considered day. When running z-*anonymity* with $z > 1$, these uncommon websites are not released, as they are associated with less than z for most of Δt. Focusing on the orange dashed line for $z = 10$, starting from the 200^{th}-ranked website, the probability of observing it in the released data falls rapidly (notice the log scale). Higher values of z (green and red dashed lines) result in earlier and steeper decrease of p_a^Y. I can compare this figure with Figure 3.1, which

[4] In case of HTTP transactions, the domain name is extracted from the `Host` HTTP header, while in case of the TLS from the SNI header in the Client Hello message.

[5] https://www.alexa.com/topsites

[6] I opt to extract directly the p_a^X rather than λ_a since these were directly available in the collected data.

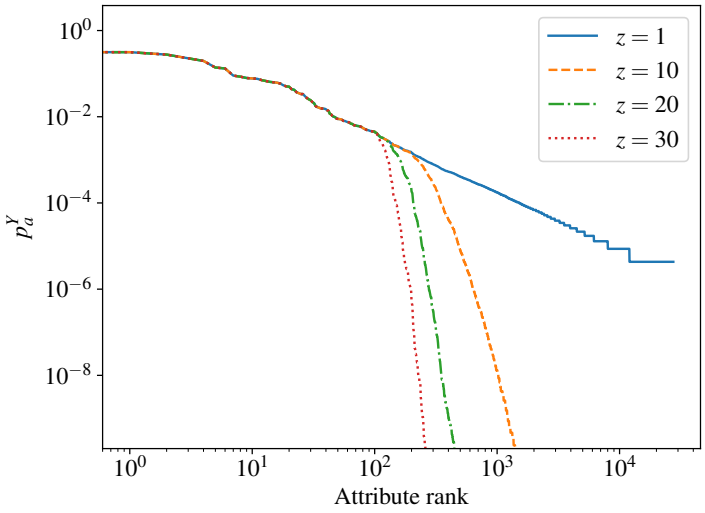


Fig. 3.6 The probability p_a^Y to publish attribute a in a $\Delta t = 1\,hour$, according to its rank, as estimated from the users' navigation data ($U = 9670$, $A = 27482$).

shows the same results for the previous case. I first notice that the dashed lines (for $z > 1$) move away from the solid blue line ($z = 1$) in the same range $10^2 - 10^3$. Secondly, I notice that the top-ranked attributes have higher p_a^Y than the previous case, with 70 websites having $p_a^Y > 10^{-2}$. This is a peculiarity of the web ecosystem, characterized by a few tens of very popular websites, including popular search engines, news portals and productivity suites, and a long tail of niche websites. In the following, I show that *z-anonymity* also works for this scenario, despite the large number of popular websites boosting the number of possible attribute combinations.

I now evaluate the impact of *z-anonymity* on the released data in terms of the k-anonymity property. Running the probabilistic framework described in Section 3.3, I can derive the probability p_{k-anon} that a given user has at least $k-1$ other users with the same attribute set. I show the results in Figure 3.7, where I report how p_{k-anon} varies with z, for different values of k. Focusing for example on the blue solid line (for $k = 2$), I notice that z must exceed 200 for p_{k-anon} to move away from 0. p_{k-anon} reaches 1 when z is 350. When considering higher k (dashed lines), larger z are necessary for p_{k-anon} to get close to 1. However it is not necessary a drastic increase of z; for $k = 4$ (green dashed line), $z = 380$ is already enough. Interesting is the comparison of the website visits with the previous case study in Figure 3.3: here z shall reach

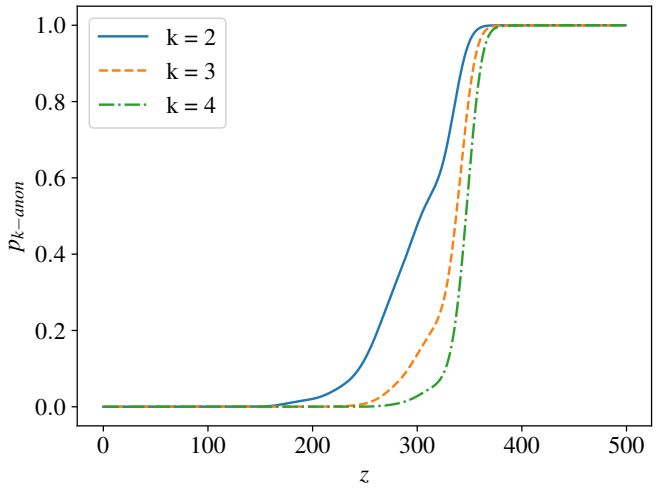


Fig. 3.7 The relation between *z-anonymity* and k-anonymity in the users' navigation data ($U = 9\,670$, $A = 27\,482$).

380 to obtain k-anonymity almost certainly, while $z = 35$ is already enough for the previous case. Two reasons are behind this. Firstly, I have only 9 760 users for the website visits, while $U = 50\,000$ in the previous case, decreasing the probability of finding users with the same set of attributes. Secondly, the probability p_a^X to expose an attribute is quite different for the two cases, with the most popular websites being visited by a large portion of users on a hourly basis. *z-anonymity* can provide reasonable guarantees of k-anonymity even in this case, provided it is properly tuned. However, this guarantees come at the cost of publishing a small number of attributes. This exemplifies the tension between data usefulness and privacy.

3.6 Limitations and future work

With *z-anonymity*, I only prevent users' re-identification if an attacker leverages uncommon attributes, by hiding z-private ones. It is designed uniquely to avoid such kind of re-identification, and, so far, I do not consider other kinds of attacks, e.g., targeting the timing or order at which users' entries appear in the data stream. Moreover, *z-anonymity* does not consider combinations of z-anonymized attributes, treating them independently. Still, I provided

a probabilistic framework that shows that users can be also k-anonymized with a controllable probability even in case an attacker knows the entire set of released attributes. With this, I provide guidelines to properly tune the system parameters to also guarantee k-anonymity. This allows the data curator to understand the properties of the released data and manage the trade-off between privacy and data utility.

Future work goes in manifold directions. First, our probabilistic framework can be employed not only to assess how *z-anonymity* results into k-anonymity, but also to *dynamically* tune z to achieve a desired k. The probabilistic framework assumes all users behave the same. Clearly, this is a simple and strong assumption and it can be refined considering classes of users with different rates of activity as well as diverse behaviors. Moreover, in *z-anonymity*, I only considered blurring z-private attributes. Alternatively, I could generalize the attributes so that they pass the z-threshold. For example, I could generalize a website to its second level domain or its content category. Moreover, I argue that I can achieve better data utility while avoiding users' re-identification at the same time even if some z-private items are released. This can be obtained by introducing perturbations in the released data, e.g., by inserting noise in the data stream or modifying some of the associations between users and attributes. Such an approach melds concepts from the classical k-anonymity with the ideas of differential privacy, where the addition of noise is the means to achieve users' privacy. All this is going to be addressed in [64], where further studies are underway to improve the model and study its full potential.

3.7 Conclusion

In this chapter, I presented *z-anonymity*, a novel anonymization property suitable for data streams. I designed it to operate with high dimensional data, organized in transactions (atomic information about users) and with the constraint of zero-delay processing. The idea at the base of *z-anonymity* is to hide z-private users' attributes, i.e., those associated with less than $z-1$ other users, which could be used by an attacker for re-identification. I show that *z-anonymity* can be achieved with an efficient algorithm if using suitable data

structures. A data stream undergoing *z-anonymity* is immediately anonymized and is available with zero delay to the consumer.

z-anonymity is weaker than k-anonymity, as it operates on users' attributes independently without considering their combination. However, I provided a probabilistic framework to map z-anonymity into k-anonymity, using which the data curator can tune the trade-off between privacy and data utility. I show a practical use case, in which I evaluate *z-anonymity* using the characteristics of a real dataset of users accessing websites. I show that it is possible to tune the system parameters to obtain k-anonymity with a controllable probability also in this scenario.

Chapter 4

Analysis of Campus Traffic during COVID-19 Pandemic

The previous chapters presented the tool that allowed me to obtain anonymous information on the functioning of the e-Lerning platform of *Politecnico di Torino*. In this chapter I present *Campus traffic and e-Learning during COVID-19 pandemic* published in the journal *Computer Networks* [65].

4.1 Introduction

Since its first outbreak between late 2019 and early 2020 in China, the COVID-19 pandemic has had a massive impact on people's lives and habits. The countries most affected by the virus spreading are facing an unprecedented health crisis, whose effects will impact their economic and social structures for a long time. The urge to respect social distancing and lockdown measures adopted to limit the spreading of the infection led to a shift in the fruition and supply of a wide number of services. Some examples include the increased usage of home delivery services, the shift to online lessons and the adoption of remote working solutions.

Italy has been among the first countries hit by COVID-19. The first case was identified in the north of Italy on February 21^{st}, and on the same date, the Government issued the fist law decree to impose quarantine in small selected towns. On February 25^{th}, the Government extended the restrictions to impose

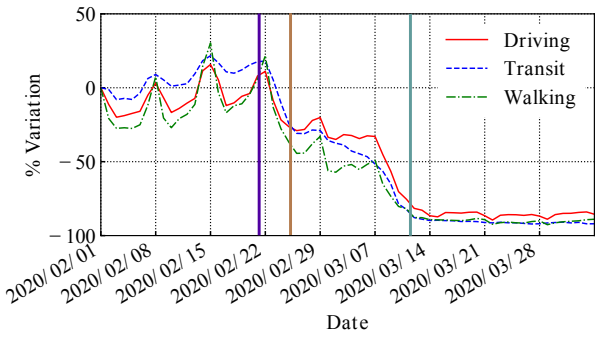


Fig. 4.1 Variations in the mobility patterns in Italy between February and April 2020. Colors mark the days in which new Ministerial Decrees introduced mobility restrictions. Source: https://www.apple.com/covid19/mobility.

remote working for all public offices, shutting down schools, and classes at Universities. Restrictions applied to the largest four regions in the north of Italy. On March 1^{st}, these restrictions were extended to the whole Italy, with further lockdown actions entering in place on March 4^{th} and 8^{th}. On March 11^{th}, the "#IoRestoACasa" Prime Ministerial Decree imposed a total lockdown to the whole Italy. Since then, and still at the time of writing, people are allowed to exit from home only for specific and urgent needs. Common retail businesses, catering and restaurant services are suspended. Gatherings in public places are prohibited. All activities not deemed essential for the Italian production chain are closed. Italy entered the most restrictive lockdown in its history.

Figure 4.1 clearly depicts the impact of these measures.[1] It shows the variation of the mobility patterns in Italy since the begin of the emergency and the impact of the restrictions and total lockdown. The vertical bars highlight the relevant events listed above. The leftmost violet bar identifies the date of the first COVID-19 case in Italy. The central brown bar identifies the school shutdown. The rightmost bar marks the date of the "#IoRestoACasa" decree. The decrease in mobility is drastic after each of those events.

Restrictions limited people's mobility while remote working, e-learning, online collaboration platforms started to grow along with online leisure solutions, like gaming and video streaming. These new habits highlighted the fundamental role of the Internet. Correspondingly, Internet traffic volume has grown by about

[1]Source: https://www.apple.com/covid19/mobility

40%, sometimes with a decrease in the download performance, questioning the resiliency of the Internet itself [66, 67].

4.1.1 My Contribution

In this chapter, I analyze the changes in the traffic patterns that are visible from my university campus, the Politecnico di Torino (PoliTO for short) in Italy. I look at the campus traffic, focusing on collaboration and remote working platforms usage, remote teaching adoption, and look for changes in unsolicited/malicious traffic. PoliTO opted to implement an in-house e-learning solution based on the BigBlueButton framework to support all the classes of the second semester, which were scheduled to start on March 2^{nd}.[2] The platform has been designed, installed, and tested during the first week of March, going live for the starting of the online semester one week later. Here I leverage this unique point of view to observe changes in the campus traffic and services during such a singular event. Moreover, I dig into details on how students access online classes and teaching material. Since students enjoy classes from their homes at different places, connected by different network operators, I check whether and how these factors affect the performance of the online teaching systems.

Overall, I highlight a 10 times decrease in incoming traffic during the lockdown. Outgoing traffic grows instead of 2.5 times, driven by more than 600 daily online classes, with around 16 000 students per day that follow classes. Online collaboration exploded, with faculty and staff members exchanging more than 17 000 chat messages and participating in more than 1 000 calls per day. I observe a surge in remote learning and working also during the weekends. Considering Internet connectivity, I notice no major problems, with only a few cases of poor performance, possibly related to people connected via 3G/4G operators.

In a nutshell, I believe this chapter testifies how the Internet proved able to cope with the sudden need for connectivity. I attest how remote working, e-learning and online collaboration platforms are a viable solution to cope with the social distancing policies during COVID-19 pandemic. While I all hope the

[2] https://bigbluebutton.org/

latter will soon be relieved, I believe the experience of online collaboration will continue also after the COVID-19 disappears.

This chapter is organized as follows: Section 4.3 describes the datasets and methodology; Section 4.4 reports a drill-down on the aggregate campus traffic patterns and online collaboration solutions; Section 4.5 focuses on online classes; Section 4.6 details network performance metrics, breaking down by operator and region; Section 4.7 looks into possible changes in unsolicited and malicious traffic, such as portscans and spam emails; Section 4.2 summarizes related work; finally Section 4.8 concludes the chapter.

4.2 Related Work

The impact of the COVID-19 epidemic on the Internet ecosystem has been immediately measured by ISPs, content providers and specialized enterprises and published in the form of reports, news and blog posts. SimilarWeb, a company that provides web analytics services for businesses, shows how the habits of people shifted during the outbreak [68], with some sectors like air travel, hotels and car rentals witnessing a large reduction of accesses while others (e-commerce, food delivery, and social networks) increased their popularity. Microsoft reported a 755% increase in usage of its cloud services [69], while YouTube and NetFlix reduced the streaming quality in Europe to prevent overload on the network [70]. CloudFlare and Fastly, two of the largest Content Delivery Networks worldwide, report a 20-40% increase in daily traffic since the lockdown in Italy [66, 67], with somehow reduced performance. Considering ISPs, the surge of network traffic has been testified with public news and blog articles. Vodafone states that fixed broadband usage has increased by more than 50% in Italy and Spain [71], with a 100% surge in upstream traffic and 44% for downstream. Telefonica IP networks experienced a traffic increase of close to 40% while mobile voice use increased by about 50% and 25% in the case of data [72]. Also European Internet Exchange Points measured an increase of traffic in the order of 10-40% [73, 74]. Outside Europe, Comcast, one of the largest US operators, reports a 32% increase in upstream traffic growth and an 18% increase in downstream traffic growth [75]. In this work, I

complement the above reports with additional figures showing the impact of the COVID-19 outbreak on the Campus networks and e-learning facilities.

Considering research papers, specific works targeting the effects of the COVID-19 pandemic have not yet been published. However, the role of the Internet has already been studied for the past catastrophic events. Cho *et al.* [76] study the impact of the 2011 earthquake in Japan on traffic and routing observed by a local ISP, while Liu *et al.* [76] focus on inter-domain rerouting analyzing BGP data. Zhuo *et al.* [77] underline the importance of the Internet during the 2011 Egyptian Revolt, while Groshek [78] finds statistical evidence that the Internet and mobile phones have helped to facilitate sociopolitical instability. Dainotti *et al.* [79] further address this aspect by analyzing the internet outages due to censorship actions that occurred during the Arab Spring revolts. Heidemann *et al.* [80] analyze network outages during the 2012 Sandy hurricane in the US. Sudden variations of Internet traffic have been observed also during massive software updates or following the born of new services (e.g., NetFlix) or protocols (e.g., Google QUIC) [11, 81, 82]. Finally, some works propose general techniques to study or improve Internet traffic during disruptive events [83–85]. The events connected with the COVID-19 pandemic have a global scale and forced an unprecedented number of people to suddenly change their habits. As such, it is of great interest to study the impact on the Internet, and this chapter provides an in-depth analysis of the Campus network traffic before and during the outbreak.

4.3 Datasets and methodology

PoliTO is a medium-sized university that offers bachelor, master and graduate-level courses in the engineering and architecture fields only. It is among the top Universities in Italy. PoliTO has about 35 000 students, of which 30% come from the Piedmont region where Torino is, 55% from other Italian regions, and 15% from the rest of the world. PoliTO employs about 2 000 faculty and researchers, and around 1 000 administrative staff members.[3] PoliTO main campus network hosts all its IT services, and offers both Ethernet and WiFi connectivity to departments, offices, classrooms, student rooms,

[3] https://www.polito.it/ateneo/colpodocchio/index.php?lang=en

and laboratories. Two 10 Gbit/s access links connect the campus LAN to the Internet, through the GARR (Gruppo per l'Armonizzazione delle Reti della Ricerca) network, which provides Internet access to all Italian universities.

In the following, I describe the data sources I use to gauge the changes in the traffic and services hosted at the campus network. If not explicitly said, the data cover the period from Saturday, February 1^{st} to Sunday, April 5^{th} 2020.

4.3.1 Passive traces

I leverage statistics collected by edge routers to observe and compare the load on different Italian University networks. This data is stored by GARR in a central database publicly accessible.[4] The repository stores the time series of the traffic volume on each edge link of the GARR network, with a granularity of five minutes. In my analysis, I consider PoliTO campus and compare it with two other large Italian universities for reference, namely Politecnico di Milano (the biggest technical university in Italy) and Università di Torino.[5]

While GARR offers only high-level aggregated data, here I also leverage on fine-grained measurements exposed by the monitoring infrastructure deployed in the Campus network. Such infrastructure is based on $\alpha - MON$ (Ch. 2, Ch. 3) that anonymizes traffic entering and leaving the Campus, that is then captured and analyzed by a passive sniffer called Tstat [62]. It computes flow-level logs similar to NetFlow, exposing information about TCP and UDP flows observed in the network. Beside classical flow-level fields, such as IP addresses and port numbers, Tstat exposes metrics such as Round-Trip time (RTT) and packet losses. I store Tstat logs in a secured Hadoop-based cluster. All data collection is approved and supervised by the responsible University officers.

4.3.2 Application logs

In addition to passive measurements, I extract data about servers and network devices offering specific services and applications that staff members

[4] https://gins.garr.it/home_statistics.php

[5] Università di Torino is a separate University which offers degrees on sciences, humanities, economics, etc.

may use for smart working. I focus on three classes of services: (i) the tools to access the internal resources of the Campus while working from home, namely Virtual Private Network (VPN) and Remote Desktop Protocol (RDP) services; (ii) the Microsoft Teams collaborative platform, which PoliTO adopted since 2019 to offer chat, file sharing, video calls and collaboration services to employees and students; (iii) email, antispam and security services. Logs available on the management consoles offer rich information about the usage of these services over time and let us study how the adoption of such services varied during the epidemic.

4.3.3 Virtual classrooms

To face the lockdown during the COVID-19 outbreak, PoliTO opted to set up an internally-hosted virtual classroom service for all classes. The system is based on the BigBlueButton framework for online learning, with customization to integrate it in the existing teaching portal. A total of 41 high-end servers running the BigBlueButton components and hosted in the Campus data center were set up in the first week of March. All classes started a week late. The BigBlueButton client-side application is based on HTML5. It provides high-quality audio, video and screen sharing application using the browser's built-in support for web real-time communication (WebRTC) libraries. In a nutshell, once the virtual classroom has been set up, BigBlueButton servers act as WebRTC Selective Forwarding Unit (SFU) capable of receiving multiple media streams (video, audio, screen sharing, etc.) and deciding which of these media streams should be sent to which participants. Video is encoded using the high-quality open-source codec VP8. PoliTO's setup lets the lecturer choose four video quality levels with a bitrate of 50, 100 (the default), 200, 300 kbit/s. Screen sharing may require the highest share of bandwidth, and, if the presenter's screen is updating frequently, the BigBlueButton client could transmit up to 1.0 Mbit/s. Audio is encoded using the OPUS encoder at 40 kb/s.[6] Finally, sharing slides takes little bandwidth beyond the initial upload/download of the pdf file.

In my analyses, I collect and process the application logs of the video servers to study the consumption and performance of virtual classrooms. I also make

[6] https://docs.bigbluebutton.org/support/faq.html#bandwidth-requirements

use of the logs of the legacy teaching web platforms to study access to the teaching material like lecture notes, pre-recorded classrooms, etc.

4.3.4 Security monitors

At last, I analyze logs from the campus border firewall that limits the access towards authorized servers while blocking possibly malicious traffic and well-known attacks. Since I am interested in remote-working solutions, I check the firewall for alarms related to SSH, RDP, SIP attacks. Intuitively, I want to check if the attack patterns changed during the COVID-19 pandemic.

The firewall is configured to let three /24 subnets to be completely open to the Internet, with no hosts connected to it. These sets of addresses act as "darknets", i.e., sets of IP addresses regularly advertised which do not host any client or server. Any traffic the darknets receive is unsolicited by definition [86]. By passively analyzing incoming packets, I observe important security events, such as the appearance and spread of botnets, DDoS attacks using spoofed IP address, etc. I use this information to further quantify if there is any change in attack patterns to my campus network during the pandemic.

4.3.5 Ethics

In this scenario, re-identification is a problem that must be taken into consideration: as said previously I took advantage of many sources of data that may contain critical sensitive information. Since I have full control of the infrastructure (recall Fig. 1.1 for reference), I can ensure impossibility of re-identification because all datasets are anonymized thanks to α-*MON* (Ch. 2, Ch. 3). Here, application level information are discarded, so the z-anonimity module is not active: hence I take advantage of anonymization up to transport level. In this way I handle data without knowing any correlation between source and action.

Regarding the data collected from external sources, re-identification is not possible since they are already aggregated.

4.4 Impact on campus traffic and remote working solutions

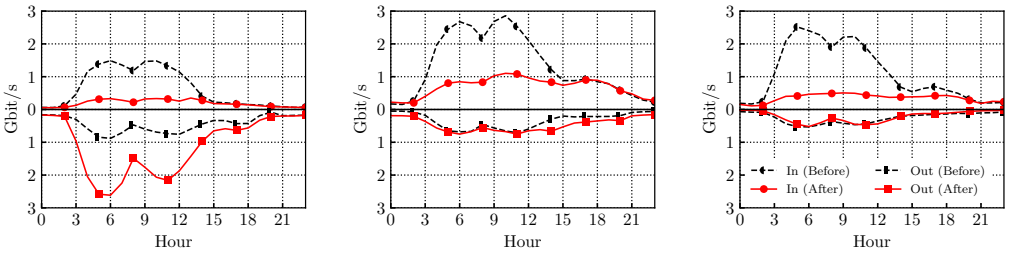
In this section, I analyze the impact of the COVID-19 outbreak on campus networks. I first provide quantitative figures on the overall traffic volume changes. Then, I focus on services supporting smart working.

4.4.1 Aggregate traffic volume

I first focus on the volume of traffic entering and leaving three Italian university campuses. Figure 4.2 shows the average hourly bitrate. For each hour, I compute the average bitrate seen over five working days. The positive y values represent incoming traffic (traffic directed to clients and servers hosted in the Campus LAN), while negative y values report the outgoing traffic (traffic directed to clients and servers on the Internet). Lines mark the volumes before and after the lockdown: the black lines depict the average per-hour bitrate observed during the week before the lockdown (third week of February), the red ones refer to averages calculated on the second week after the lockdown (third week of March). For comparison, the figure includes plots for the Politecnico di Torino (PoliTO) in Figure 4.2a, the Politecnico di Milano (PoliMI) in Figure 4.2b and the Università di Torino (UniTO) in Figure 4.2c.

Focusing on the positive $y-$axes, observe how the incoming traffic has shrunk in the three cases, reflecting the lockdown effects. Since the second week of March, most students, researchers and staff members cannot access the campuses. The traffic after the lockdown is about one tenth of the traffic before it in both PoliTO and UniTO, with PoliMI still observing some sizeable incoming traffic. This difference reflects the different lockdown policies imposed by each university. PoliTO and UniTO completely blocked all teaching and research activities, while PoliMI still allows the activity of some laboratories.

The negative $y-$axes report the outgoing traffic. Again, the three campuses present different behaviors: I see a major increase in outgoing traffic from PoliTO, which is not observed in other campuses. This behavior is justified by the online teaching platform hosted in PoliTO which causes an increase of about 2.8 times the baseline outgoing traffic during peak time. PoliMI and UniTO



(a) Politecnico di Torino. (b) Politecnico di Milano. (c) Università di Torino.

Fig. 4.2 Traffic from three Italian Universities before and after the lockdown.

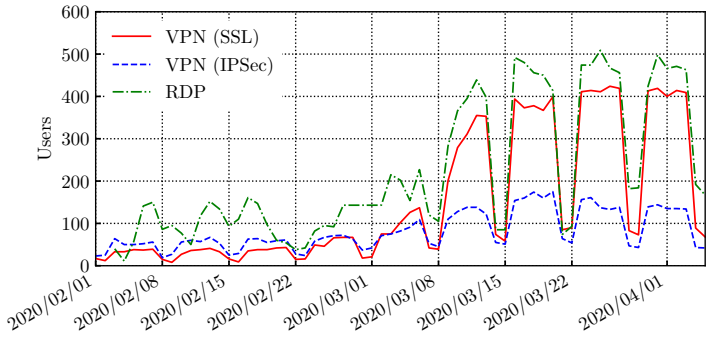


Fig. 4.3 Daily accesses to in-house PoliTO smart working systems.

have resorted to cloud-based solutions, hence the outgoing traffic volume does not show relevant changes before and after the lockdown.

Takeaway: Campus incoming traffic drastically reduced during lockdown. Outgoing traffic changed in PoliTO, where an in-house online teaching service has been deployed. This system caused a massive inversion on traffic patterns, with significant growth in upload traffic due to online teaching services.

4.4.2 Smart working adoption

I now focus on PoliTO only and drill down on the services used by the personnel for working remotely.

In Figure 4.3 I show how the number of users relying on VPNs and remote desktop (RDP) solutions have changed over time. VPNs are used to access

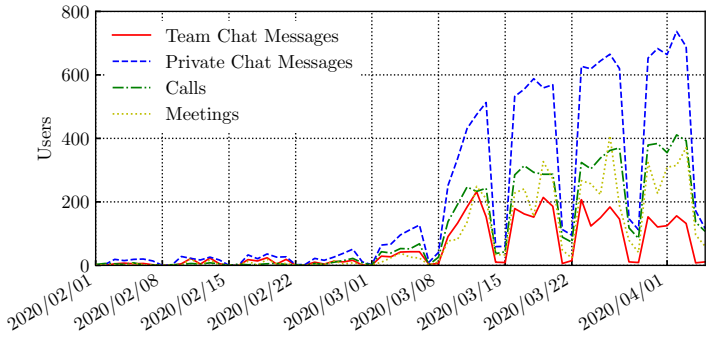


Fig. 4.4 Daily activity on PoliTO Microsoft Teams systems.

in-campus services and servers, while RDP allows one to remotely access files or run applications on their office computers. Curves in Figure 4.3 depict the total number of unique users accessing the services at least once in a day.[7] The effect of the lockdown started on March 11^{th} 2020 is astonishing. Since the lockdown started, these solutions simultaneously present a relevant increase in usage. PoliTO offers VPN services both over IPSec and SSL. Interestingly, SSL-based VPN usage increases significantly more than the IPSec-based solution. This suggests that the lockdown forced non-expert users to resort to a VPN, and they have opted for SSL-based VPN, which is easier to configure.

Users working from home also heavily rely on RDP, with about 500 users contacting such services at least once in a day after the lockdown. Sessions (not reported for the sake of brevity) last several hours, suggesting that this remote access method is mostly used for regular working sessions, and not only to access files on computers left in offices. Notice also the growth in the number of accesses over weekends, with more than 200 RDP accesses per day. This suggests that people, forced at home, keep working during the weekend, too.

[7]VPN accesses are directly extracted from the VPN terminator logs. I identify RDP connections from Tstat logs, considering TCP flows directed to port 3389 that exchanged at least 10 MB of data. I count users by their client IP addresses, which I assume to be unique on a daily basis.

4.4.3 Remote collaboration

I now move to remote collaboration suites. PoliTO offers all employees and students free access to the Microsoft Teams platform for smart working. Running on the cloud, it puts little loads on the campus network. Yet, its usage provides insights on people's habits during the lockdown.[8] Figure 4.4 summarizes the activity of Microsoft Teams users. It shows the number of *team chat messages* (i.e., chat messages to groups), *private chat messages*, *calls* and *meetings*. In Teams' jargon, both *calls* and *meetings* are online video conferencing; *calls* have two participants.[9]

Each point in Figure 4.4 marks the number of people using each functionality at least once in a day. Teams was already available but only marginally used in the campus before the lockdown, with few tens of users per day. As for other services supporting smart working, its usage explodes during March. Compared to smart working connectivity solutions – see Figure 4.3 – the growth is slightly less abrupt, showing that the people resorted first to means to access their data and office computers, and then to online collaboration tools. People rely on Microsoft Teams as a means to exchange direct messages – more than 700 daily users sent private chat messages over Teams in the last week of March. In total, they exchanged more than 17 000 messages per day. Interestingly, video calls and meetings keep growing as well, topping to 400 users per day, making more than 1 500 calls per day. Team chat messages instead show a drop in popularity after an initial surge.

Takeaway: The lockdown has pushed smart working to widespread adoption. Remote services access via VPN and RDP, VoIP communications and online conferencing suddenly become regular for PoliTO users.

4.5 Online teaching

In this section, I study in detail the fruition and the performance of the online teaching system deployed at PoliTO. The compelled moving of all classes

[8] PoliTO staff members are not compelled to use Microsoft Teams, I, therefore, expect other platforms to show similar significant growth.

[9] https://docs.microsoft.com/it-it/microsoftteams/teams-analytics-and-reports/teams-reporting-reference - accessed April 2020

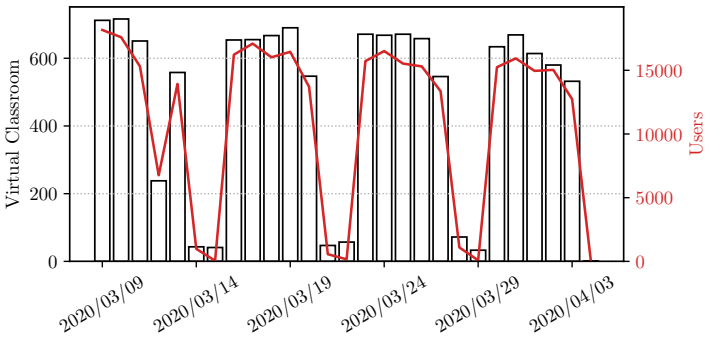


Fig. 4.5 Daily number of virtual classrooms and connected students (faculties and students).

to an online solution allows us to evaluate (i) the impact of such a scenario on the campus networks, (ii) how students accessed these new services, and (iii) if students suffered any impairment due to limitation in the remote connectivity.

4.5.1 Audience

Figure 4.5 provides a summary of the audience of PoliTO's online teaching system. Only March is shown since the university has deployed the system from scratch to cope with the emergency.

The number of virtual classrooms (black bars) follows the pattern of the university lecturing schedules, with around 700 virtual classrooms per day. The drop registered on Thursday of the first week was caused by an outage. More than 16 000 students connect to at least one virtual classroom on a daily basis. Considering that around 35 000 students are registered at the university, and some join a few lectures a day, a large percentage of the community engages in online teaching each day. The trend on active classrooms and students is stable over time, suggesting that students and faculty found the virtual teaching platform appropriate. This is confirmed by the feedback students leave at the end of each class which sees more than 70% of students giving a feedback of 4 or 5 stars (5 being the highest rating) to the technical quality of the session.

Interestingly, 47% of classes have been followed by students outside Piedmont, where PoliTO resides. Some students have indeed returned to their

home countries or regions before the start of the second semester and could not return to Torino because of the lockdown. Other statistics show that most students follow classes using a Windows 10 PC (64.5%) or Mac OS X (14.1%), and Chrome (71%), Safari (10.1%) or Firefox (9.9%) browser. Yet, 6.2% and 2.4% of students follow the class from an Android or iOS device, respectively.

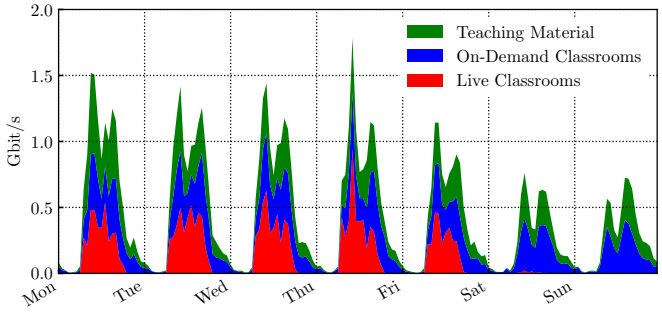
Takeaway: Engagement in online teaching is impressively high during the COVID-19 emergency, with a large part of the academic community participating on a daily basis. Virtual classrooms allowed students to follow classes, with almost 50% of them connecting from outside Piedmont.

4.5.2 Network workload

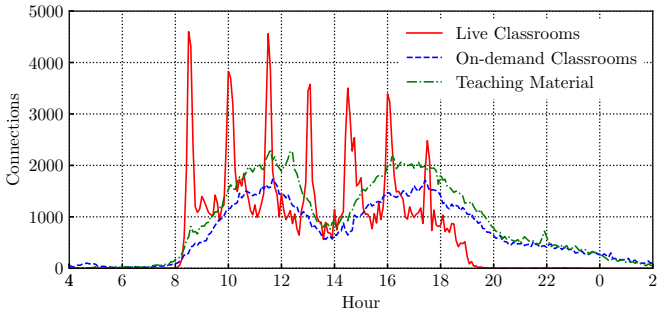
How much massive online teaching costs in terms of network traffic? Clearly, the answer to this question depends on the technical parameters of the online teaching environment. PoliTO's infrastructure includes the deployment of teaching tools for (i) virtual classroom; (ii) on-demand video of pre-recorded classes (stored as 720p video); (iii) other teaching material (e.g., slides, teachers' notes, other useful files). The virtual classroom service allows the students to watch a live video of the lecturer, an optional whiteboard as well as the direct sharing of the lecturer's screen. The BBB platform uses dedicated servers and WebRTC to set up multiple RTP streams that carry the live video and audio. On-demand video and teaching material are instead offered by additional servers as standard HTTP downloads.

Figure 4.6a details the traffic volume for each service during the fourth week of March. The figure reports the average bitrate of the traffic leaving the Campus – i.e., from the servers to the clients, for each hour. During weekdays, the total bitrate exceeds 1 Gbit/s, with live streaming of classes responsible for a bit more than one third of the traffic. During the weekend, when no lecture is scheduled, I still observe large traffic (up to 750 Mbit/s) due to students downloading on-demand lectures and teaching material. Virtual classroom traffic starts and ends within the schedule, while students keep accessing on-demand classroom and material also at night.

Figure 4.6b shows how the accesses to teaching facilities are distributed over the day. I take the start time of each student connection to a teaching



(a) Weekly.

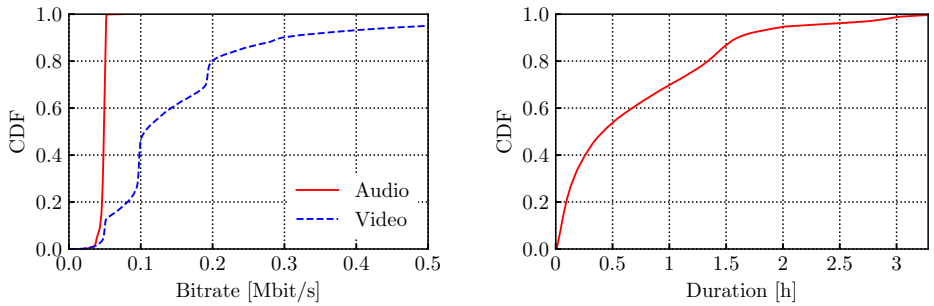


(b) Daily.

Fig. 4.6 Access pattern to the teaching services.

server and plot the number of new connections observed every five minutes. Accesses to live classrooms (red solid line) clearly follow the schedule of the campus lectures, which begin every 90 minutes, from 8:30 AM until 7 PM. When lectures begin, thousands of students start a new session with the live classrooms servers, resulting in peaks of more than 4 500 connections. Different is the case for on-demand classrooms and teaching material (blue and green dashed lines, respectively), whose consumption is spread over the day. As said above, students download such material also late in the evenings, including sizeable accesses even after midnight.

I complement the above picture with Figure 4.7, in which I show different characteristics of the virtual classroom sessions. Using logs exported by Tstat, I analyze the RTP streams that serve multimedia content (audio and video) to the students and report the distribution of the per-flow average bitrate in



(a) Bitrate.

(b) Duration.

Fig. 4.7 Characteristics of virtual classroom sessions.

Figure 4.7a, separately per audio and video.[10] All classes carry an audio track, with 40 kbit/s average bitrate – the expected VoIP bitrate according to the BBB configuration. Considering video, I observe that lecturers usually select the medium quality at 100 kbit/s, but I observe a considerable amount of streams at 50 kbit/s (low quality) and 200 kbit/s (high quality). Remind that PoliTO setup allows four quality levels, as explained in Section 4.3.3. In only 10% of the cases, streams reach 300 kbit/s or higher, meaning that ultra-high quality video and screen sharing are seldom used. In all cases, bandwidth is not a major problem for BBB, and students need no more than 0.5-1.0 Mbit/s to enjoy a lecture.

At last, I consider session duration in Figure 4.7b. Half of the streams last less than 30 minutes, but it is hard to link this short duration to abandonment by students, as the server could reset multimedia streams for other reasons (e.g., the lecturer temporarily mutes the microphone, or takes a break during a class). Interestingly, I still notice two bumps at 1.5 and 3 hours, which are the typical duration of PoliTO classes.

Takeaway: Live classrooms put a large workload on teaching servers with significant peaks on scheduled lecture times. Yet, bandwidth is not a major problem. Students need less than 1 Mbit/s downstream bandwidth to fully

[10]RTP streams are defined by the combination of endpoint addresses and port numbers, plus the SSRC stream identifier. I distinguish audio and video using the RTP Payload Type field.

enjoy live lectures. Not-live teaching media are heavily consumed also during evenings and weekends.

4.6 Network performance

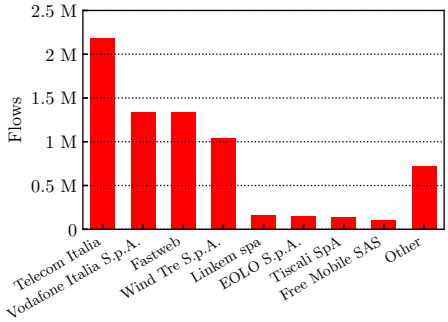
I now evaluate network performance metrics to understand whether people accessing teaching material experience problems to obtain the content. I first investigate the performance by breaking down data according to the students' Internet Service Providers (ISPs). Then, I break down figures according to the geographical region from where students connect. In both cases, I map their ISPs and regions using the MaxMind datasets.[11] Results in this section are computed by considering the downloads of on-demand video or teaching material with TCP connections. Both contents are provided through HTTP bulk transfers and thus constitute a valid download performance test. To reduce noise, I consider only downloads of objects of at least 10 MB.

4.6.1 Internet Service Providers breakdown

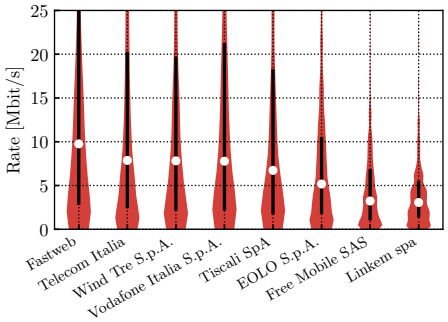
Figure 4.8a shows the distribution of the number of connections according to the ISPs hosting the client IP addresses. As expected, the largest 4 ISPs in Italy dominate the list, with TIM grabbing 31% of the connections, followed by Vodafone, Fastweb and Wind Tre getting 20%-15% of the share. Some ISPs rely on specific access technologies. For instance, EOLO, Linkem, and Free Mobile offer Internet over 3G/4G technologies only. Would students using them suffer eventual impairments?

For this, I check the distribution of download throughput for each ISP. Violin plots in Figure 4.8b depict results. Each violin plot represents the probability density function of the average per-flow throughput; White dots mark the median values of distributions. I sort providers by this median. Most flows experience average throughput higher than 5 Mb/s. For example, the median on Fastweb customers (a provider offering mostly fiber access solution) is as high as 10 Mbit/s. Figures for 3G/4G-only providers are, instead, below 5 Mbit/s, thus pointing to possible client-side impairment (e.g., congestion).

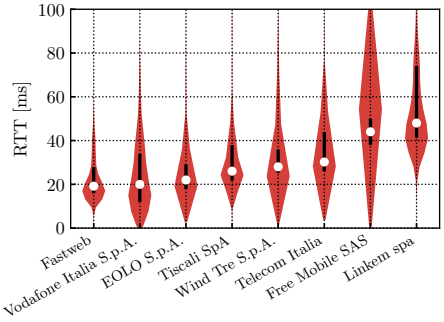
[11] https://www.maxmind.com/



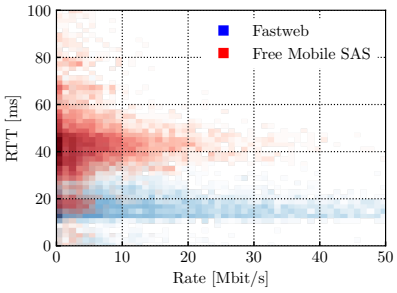
(a) Number of download sessions



(b) Download throughput distribution



(c) RTT distribution



(d) RTT vs throughput: Fastweb and Free

Fig. 4.8 Access to teaching servers per ISP. Only flows larger than 10 MB are considered.

Figure 4.8c provides more insights. The violin plots depict the distributions of the minimum RTT which Tstat computes by measuring the time between a data segment and its corresponding acknowledgment. RTT is impacted (i) by the physical distance from clients to servers, (ii) by access technology delays and network congestion, and (iii) Internet routing. Notice how the RTT distribution is very condensed for Fastweb (median at around 20 ms). On the other extreme, wireless-only providers have widespread distributions, with a median over 40 ms, and peaks above 100 ms (see Free Mobile as the clearest example). Figure 4.8d extends the analysis showing a per-flow comparison of RTT versus throughput for two providers. The low RTT for the (fiber-most) customers on Fastweb comes together with a large throughput. The more variable 3G/4G network on Free Mobile results in higher and more distributed RTT and lower throughput. In extreme cases, very large RTT values together with very low throughput are symptoms of possible network congestion.

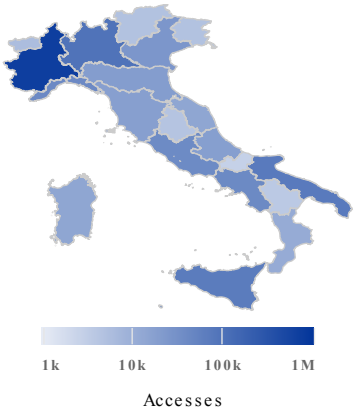


Fig. 4.9 Access to teaching material per Italian region. Only flows larger than 10 MB are considered.

Takeaway: Despite the variability of results, the very large majority of Italian ISPs can easily meet the roughly 1 Mbit/s required to enjoy live-streaming classes. Only a few customers, in particular those relying on 3G/4G access technologies, may suffer large delays and limited throughput with potential to disturb the user experience.

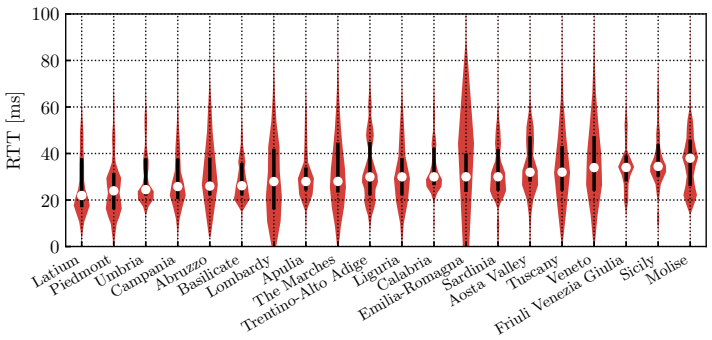
4.6.2 Geographical characteristics

I repeat the performance analysis considering the geographical region from where students connect to the online teaching system. Students in some countries[12] are reported to have problems following online lectures due to poor Internet connectivity. Whereas I cannot measure the number of students suffering from a total lack of connectivity, I can estimate whether PoliTO's students and faculty experience different performance when connecting from different regions of Italy.

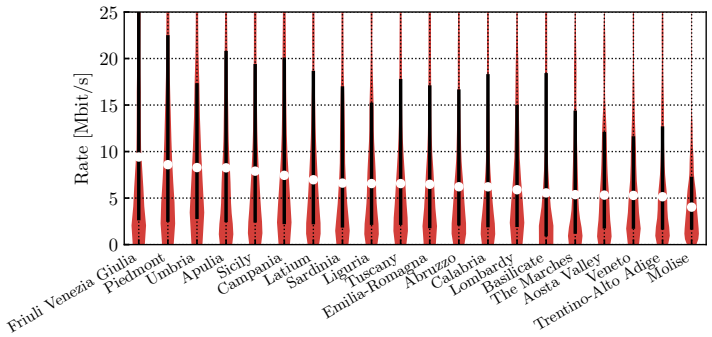
Figure 4.9 visually reports the number of flows per Italian region. Again, I consider only connections downloading at least 10 MB. I can see that people have connected to the teaching system from all Italian regions. Naturally, a larger number of connections is seen for Piedmont, where PoliTO is located –

[12] https://www.theguardian.com/commentisfree/2020/mar/23/us-students-are-being-asked-to-work-remotely-but-22-of-homes-dont-have-internet

the region marked in dark-blue in northwestern Italy. Still, significant numbers of students are seen in other regions, allowing us to make fair comparisons.



(a) RTT distribution



(b) Download throughput distribution

Fig. 4.10 RTT and throughput distributions (flows larger than 10 MB) per Italian region.

Figure 4.10 compares performance metrics for connections coming for the several Italian regions. I use violin plots once again, and sort regions in the $x-axes$ from the best to the worst median value for the metric in each plot. Figure 4.10a presents the distributions of RTT. Here physical distance is expected to play a key role. Yet, interesting patterns emerge. First, note that connections from Latium (the region where Rome is located) typically experience lower RTT than connections from Piedmont. This is an artifact due to Internet routing: Most Internet providers peer with GARR via Internet exchanges in Rome or Milan. This aspect causes routing detours that artificially increase RTT for all regions, but Latium and in part Lombardy. Second, whereas southern regions are penalized by their distance to PoliTO, much

noisier RTT distributions are seen for some northern regions. Observe, for example, that Veneto and Emilia-Romagna, two economically strong regions in northern Italian, have a large portion of connections with larger RTT than the median connection from Sicily (an island in southern Italy). This reflects again the different mix of access technologies and Internet routing of ISPs.

Figure 4.10b depicts distribution of the average connection throughput. I can see a large difference in throughput for the different regions. Connections coming from the best-performing regions have median average throughput twice as high as the regions with the lowest figures. Observe how some regions with large median RTT, such as Sicily and Friuli-Venezia-Giulia, still present good throughput figures. These results suggest that their large RTTs are mostly due to physical and routing distances rather than congestion. In some cases, large RTT variations, such as those observed for Veneto, are coupled with low throughput figures, which result in low Internet quality for a sizeable percentage of customers.

Takeaway: While some people reported limited Internet performance due to traffic increase after COVID-19 lockdown [66, 67], my data show that overall performance is still good to access online teaching. Physical/routing distance has little impact in general, with Internet access technology that is still critical for flow performance.

4.7 Security events and unsolicited traffic

Finally, I check events visible from PoliTO's network security monitoring solutions. My goal is to gauge evidence of possible changes in malicious network activities during the lockdown.

In Figure 4.11 I report the variation on the number of events reported by the university firewall per week. The first week is taken as a reference. The three most common events are shown, namely scan/brute-force attacks against SIP, RDP and SSH. The numbers for SIP and SSH show little variation throughout the weeks. These small variations are in line with the normal operations of this firewall. For RDP, some significant changes appear in the week of March 16^{th}. Indeed, the number of RDP events reported by the firewall has more

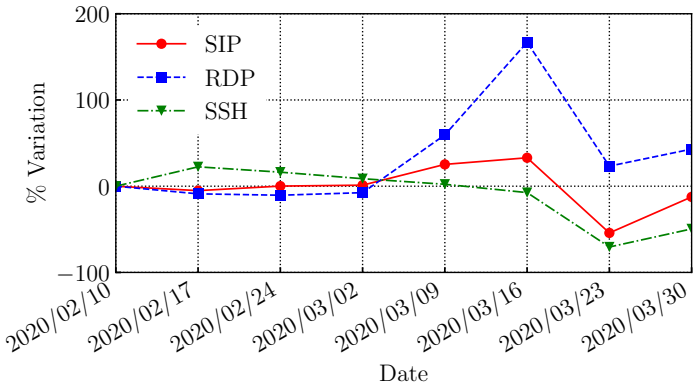


Fig. 4.11 Variations in the number of events reported by the university firewall between February and March 2020.

than doubled in the first week of lockdown. Recall that many RDP servers were enabled during that week for easing smart working. Yet, this increase is a consequence of the deployment of the new systems and would happen even without the lockdown. As potential attackers have found new RDP servers online, they perform more scanning and brute-force activity against these nodes, too. The posterior decrease in RDP events is explained by changes in the firewall setup, performed to limit such scan activity.

A close inspection of traces of the darknet deployed at PoliTO leads to similar remarks. Figure 4.12 shows the variations in the number of packets reaching the most contacted ports of the darknet. Traffic reaching the darknet is always noisy and great variations are normally seen when new large-scale Internet scans are performed [86]. I observe precisely this typical pattern during the weeks of lockdown as during any other period of time. I indeed observe episodes of sudden increases for some particular ports (e.g., port TCP/23 used by Telnet). However, there is no evident correlation among these variations and the lockdown itself.

At last, I check the load on email servers, but I omit figures for brevity. I observe a clear decreasing trend in SPAM emails. While the trend seems more prominent during March, the decrease has started months before. Again, there is no evidence that the reduction in SPAM would be a consequence of the lockdown.

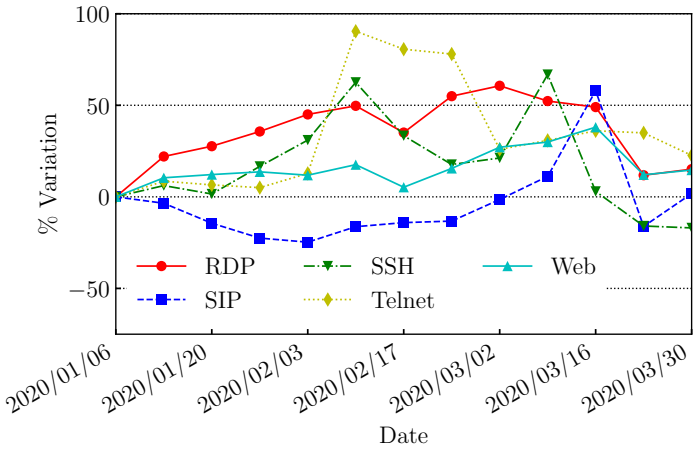


Fig. 4.12 Variation in the darknet traffic between January and March 2020.

Takeaway: I observed little changes in unsolicited traffic, SPAM email and other security-related events. There are no clear signals that the most significant events are a consequence of the lockdown.

4.8 Conclusions

To mitigate the spread of the COVID-19 pandemic, the world issued severe restrictions like social distancing and lockdown measures. This forced people to change their habits and pushed them to online services for learning, smart working and leisure, generating an unprecedented load on the Internet. Since March 11^{th} and still at the time of writing, Italy is facing a total lockdown, with 80-90% of people forced to stay at home.

In this chapter, I took the perspective of the Politecnico di Torino campus, analyzing the changes in traffic patterns due to the lockdown measures and the switch to online collaboration and e-learning solutions. I observe that incoming traffic drastically decreased, while outgoing traffic has more than doubled to support online learning.

Remote working and online collaboration exploded as well, with hundreds of staff members using the Microsoft Teams collaboration platform, VPN and remote desktop services to keep working from home. Since then PoliTO's in-

house e-teaching infrastructure is serving more than 600 daily classes to more than 16 000 students, generating peaks of 1.5 Gbit/s of traffic. I also looked for eventual impairment suffered by students using different ISPs, or connecting from different regions. Results show that the campus and the Internet have proved robust to successfully cope with challenges and maintain the university operations.

Chapter 5

DPI Performances Evaluation for Live Applications

Starting from this chapter, I will shift my attention to the aspects of network measurements that are more related to Cybersecurity by describing the right box (red) of the infrastructure depicted in Fig. 1.1 and the related datasets obtained. The first step consists in revisiting the concept of Deep Packet Inspection, useful for the construction of a system of Smart Honeypots. In this chapter I present *DPI Solutions in Practice: Benchmark and Comparison* presented in *IEEE Security and Privacy Workshops (SPW)* [87].

5.1 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 [88] or leverage on restrict traffic traces, which questions the applicability of such results to practical scenarios.

5.1.1 My Contribution

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 [89], Libprotoident [90], Tstat [46] and Zeek [91]. 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 goal of this is to understand if and which DPI solutions are able to operate in a streaming environment and be incorporated into active network traffic capture systems.

My 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, I share my code and the instructions to build the complete datasets used in my experiments.[1]

Next, Section 7.2 summarizes the related work. Section 5.3 introduces my datasets and methodology. Section 5.4 describes the results, and finally Section 5.5 concludes the chapter.

5.2 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 inspection [92], 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 [93] or finely-tailored DPI algorithms [94]. 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 [46].

Many DPI tools have been introduced implementing such techniques. Here I consider four alternatives, which have been evaluated by original authors in [90, 89, 91, 95]. 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 [96] perform an extensive benchmark covering port-based classification, packet signature algorithms etc. In [97], authors survey approaches to overcome the lack of ground truth in

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

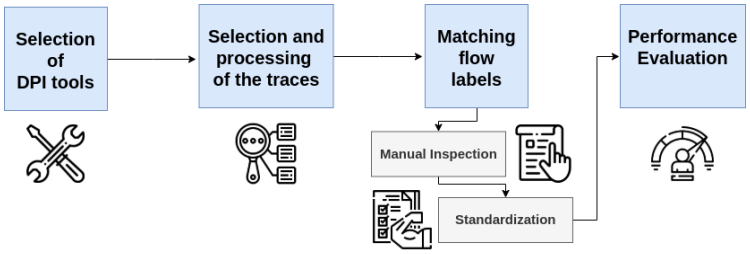


Fig. 5.1 Testing methodology.

such studies. In some cases manual labelling of packet captures is used for DPI comparisons [98], while other works rely on active measurements to enrich captures with information about underneath applications [99–101].

Closer to my analysis is the work presented in [88], where authors also provide an independent comparison of DPI solutions. In contrast to [88], I leave out of my 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.

5.3 Datasets and Methodology

Fig. 5.1 summarizes my methodology. I describe the DPI tools selected for testing (Sect. 5.3.1). Then, I build up a set of traces covering different traffic scenarios (Sect. 5.3.2). Next I process the traces with the DPI tools. As matching the obtained labels requires ingenuity, I perform several steps and build up heuristics to find discrepancies on the final classifications (Sect. 5.3.3).

5.3.1 Selection of DPI Tools

I restrict my 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.) [102, 103]. Namely, I focus on the following four alternatives:

- *nDPI* [89] 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* [104] fork optimized for performance and supports more than 100 protocols.

- *Libprotoident* [90] 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* [91] – 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* [46] 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 [98]. Equally, I ignore proprietary alternatives, given the intrinsic difficulty to evaluate the root-causes of conflicting results without access to source codes [89]. Finally, I do not evaluate *tshark*[3] as it has proved much slower than the alternatives.

5.3.2 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

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

while at home; (ii) *Media & Games* [105–107] that includes conference-calls, RTC applications, multimedia and gaming traffic; (iii) *Malware* [108], which aggregates several samples of malware[4] and security experiments;[5] and *IoT* [109, 110], captured in different labs hosting a variety of IoT devices. I include both traces captured in my 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 my results, I link the public PCAP files.[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 different numbers of flows. Thus, I need ingenuity to compare results.

Table 5.1 summarizes the number of flows reported by each tool. I see 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 my analysis, since they usually refer to flows that carry no payload. Indeed, a lot of flows without payload is 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

[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 5.1 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

artifacts related to the way flows 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.

Table 5.2 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

I report a summary of the final macrotraces in Table 5.2. 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.

Table 5.3 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

Table 5.4 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

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

5.3.3 Matching flow labels

We need some ingenuity to normalize the output of the tools and compare their classifications. First, we normalize all labels, e.g., using always lower case and removing special characters. Then, we manually verify the output strings to identify possible synonyms used across tools. Table 5.3 reports a subset of labels that require manual standardization. In total we manually evaluated 225 labels, replacing cases such as those in the right column of Table 5.3 by a single common label (left column).

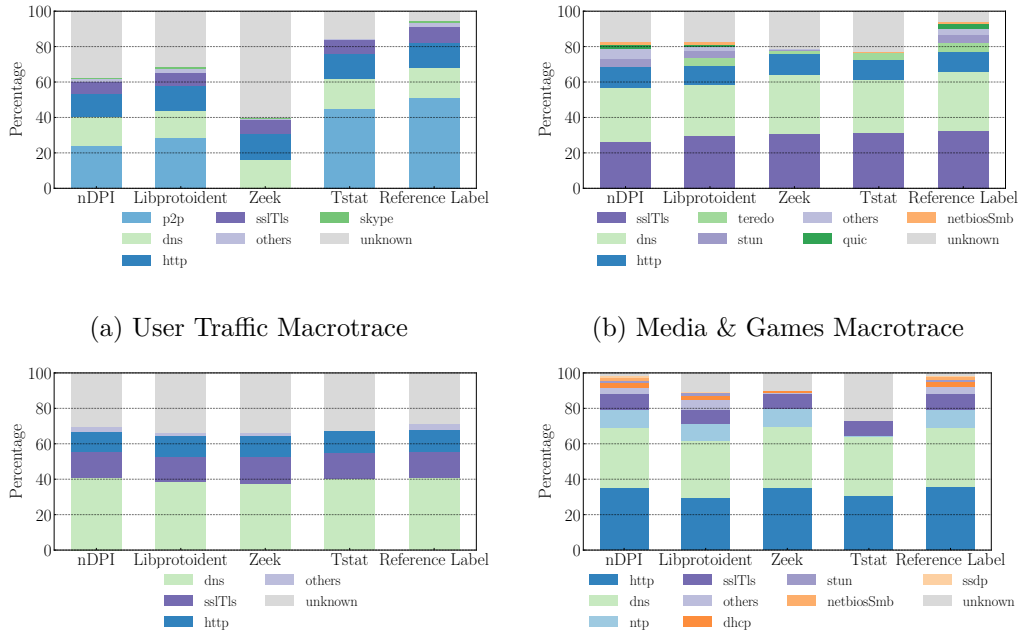
Next, we face the question on how to determine the label for each flow in absence of ground truth. Indeed, the lack of ground truth has pushed most of previous works to resort to testbeds or emulated traffic that we want to avoid [97]. We thus decide to focus on the *consistency* of different tools, i.e., we 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 5.4 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, we 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 we could not resolve are ignored, with confidence score equals to zero.

Finally, once the reference labels are defined, we calculate performance metrics for each tool. We 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.

5.4 Results

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) User Traffic Macrotrace

(b) Media & Games Macrotrace

(c) Malware Macrotrace

(d) IoT Macrotrace

Fig. 5.2 Percentage of labelled flows for each tool. The last bar in the plots reports percentages for my reference label.

5.4.1 Labelled flows per protocol

Fig. 5.2 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 my *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. 5.2b – all tools recognize close to 80% of the flows. This trace is mostly composed by HTTP, DNS and TLS traffic,

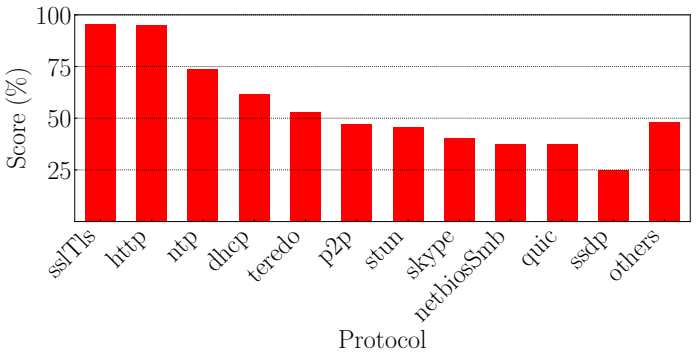


Fig. 5.3 Average per flow confidence score for the top reference labels.

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. 5.2c – 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. 5.2d – 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. 5.3 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.

5.4.2 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. 5.5. 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), I see numbers 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. As I see in the table, 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).

5.4.3 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

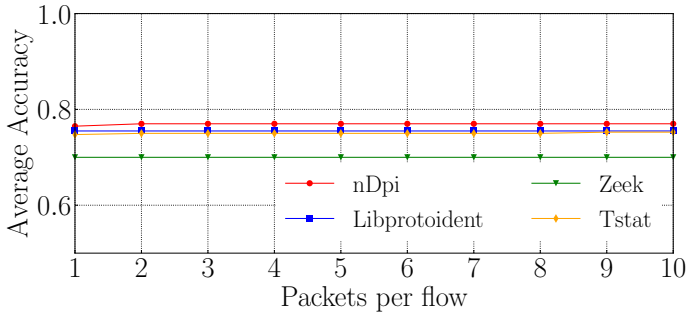


Fig. 5.4 Average accuracy when increasing the number of packets per flow. Tools reach a final classification already in the first packet with payload.

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. 5.4 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 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 my tests, all tool, but Zeek, present similar performance figures when processing a single PCAP at a time.

Table 5.5 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

5.5 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 to 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, as described in the next chapter (Ch. 6).

Chapter 6

Augmenting Darknet Capabilities through Smart Honeypots

In this chapter I will use what was discussed in the previous chapter for the development and analysis of an innovative Honeypot system, presenting *Enlightening the Darknets: Augmenting Darknet Visibility with Active Probes*, currently under evaluation at *IEEE Transactions on Network and Service Management* [111].

6.1 Introduction

Darknets or network telescopes are IP addresses advertised by routing protocols without hosting any services. They have been used for years as *passive sensors* in a variety of network monitoring activities and research projects [112–115]. Traffic reaching a darknet is inevitably unsolicited. Therefore, it is helpful to highlight network scans (both from 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 [113].

To increase the visibility of attackers' activities, *honeypots* allow researchers to obtain more information about events on darknets [116–119]. Honeypots are *active sensors* that collect information by responding to unsolicited traffic. The goal is to engage with potential attackers using simulators that replicate

the basic functions of real systems (low-interaction honeypots) or actual live systems deployed in controlled environments (high-interaction honeypots).

Darknets and honeypots are complementary: the former provides a broad but shallow view of scanning activity; the latter provides deeper insights into specific attack patterns. A combination of the two cases could enrich the type of information currently being gleaned from darknets while providing broad coverage and deep insights. Darknet traffic is known to change significantly, not only in the IP address space, but also due to production services hosted "near" the darknet's address space [120, 121].

6.1.1 My Contribution

I present my efforts to systematically and quantitatively compare different levels of interactive responders that I deploy *within* different portions of my darknet address space. I consider the following four types of sensors: (i) Darknet, silent listeners that capture received traffic; (ii) L4-Responders, which complete the TCP handshake and store all possible application layer requests sent by clients; (iii) L7-Responders, low-interaction honeypots that mimic specific application protocols on their usual and known ports; (iv) DPIpot, a novel responder that identifies the application protocol used by the sender regardless of the destination port.

I design two experiments to observe how senders[1] interact with the responders. In the first setup, I activate the responders in a /24 darknet, while keeping a second /24 network as a pure darknet. I run this setup for months and observe the traffic that each sensor attracts over an extended period of time. In the second experiment, I first turn off all responders to measure the effects of darkening a network with active services. After 15 days, I turn on responders in my second /24 darknet, measuring the transient effects of deploying active responders in a darknet.

My goal is to revisit and update some well-known facts about darknet deployments and add new and fresh insights that highlight the advantages and disadvantages of alternative response strategies. Summarizing my key findings:

[1] I call *sender* hosts contacting my darknet, e.g., attackers, scanners, etc.

- I quantify *Side-Scan* phenomena where a node hosting a service receives more traffic for other services and ports. Unlike [120], which observes similar patterns in CDN nodes, I quantify how the type of service hosted in the darknet critically affects *Side-Scan*.

- Activating responders leads to an increase in traffic on darknet neighboring IP addresses too. The joint use of active responders and passive darknets increases the visibility of sender patterns and improves the understanding of phenomena and attacks.

- L4-Responders and L7-Responders increase the visibility of darknets, as reported in [122, 116]. However, the lack of a wide range of application-level responders limits interaction and traffic visibility compared to my multiple L7-Responders and DPIpot.

- DPIpot decouples services from ports, shedding light on activities directed to non-standard ports, offering a rich picture that is unseen in other deployments. As a side-effect, it may "trap" senders on some particular activities, slowing them. This trade-off shows how passive and interactive deployments are complementary.

- I observe several scanning patterns that senders employ to discover hosts and services in a network. I document how fast hosts become the target of in-depth activity from multiple senders once found online by some initial scanners. Conversely, senders keep coming back (for weeks) to IP addresses that once hosted active responders.

In addition to these analyzes, I provide the complete set of responders used in my analysis as open-source software. I release DPIpot to foster its development and use. I also make the data analyzed in the following sections available online in anonymized form to allow for reproducibility.

I provide an overview of related work (Section 7.2), explain my methodology (Section 6.3), and describe macroscopic traffic characteristics (Section 6.4). I examine the changes in different deployments (Section 6.5) and the benefits of DPIpot (Section 6.6). Finally, I observe what happens when I darken and lighten a network (Section 6.7), before concluding the chapter (Section 7.6).

6.2 Related work

6.2.1 Darknet research and infrastructure

Darknets have been employed for years in various network monitoring and research activities [112]. Examples include the study of (i) DDoS attacks [119, 123, 124], (ii) IPv4 address space usage [125], (iii) Internet censorship [126], (iv) large-scale internet scanning [127, 115, 128], and (v) botnets and malware proliferation [117, 129].

In terms of infrastructure, previous efforts have characterized the differences between centralized and sparse implementations, size, and location of darknets [112, 114]. Several actors maintain darknet infrastructures, including the decades-old CAIDA/UCSD [130] project, darknets operated by major network operators [113, 114], and other projects run by universities and security companies worldwide [118, 121, 131–134].

Recent work [120] used servers from Akamai's Content Delivery Network (CDN) to study unsolicited traffic. Unlike a traditional darknet, CDN nodes provide public services and thus receive and process production traffic. Nevertheless, all `TCP/UDP` ports that do not host production services can be reached by unsolicited traffic. The authors show that production servers attract unsolicited traffic that is quite different from the traffic observed in an ordinary darknet. I extend these findings by uncovering and investigating different deployment combinations and services. Although my deployment is based on honeypots and thus lacks components of a production environment, I show that the combination of services exposed in a host is important and shapes the mix of attacks and noise that targets it.

Similar to my methodology, authors [122] propose Spoki, which completes TCP handshakes in the darknet to record the first payload sent by scanners. The authors of [116] present eX-IoT, IoT honeypots deployed in darknets. Spoki is similar to my L4-Responders, while eX-IoT is a new category of L7-Responders. My methodology includes multiple functions beyond Spoki and eX-IoT, including multiple categories of L7-Responders and a new DPIpot that performs deep packet inspection (DPI) on-the-fly to decide how to respond to scanners. I show that advanced responders shed light on a new wave of

scanners and attackers that are not visible in a pure darknet or when using a single responder type such as in [122, 116].

6.2.2 Honeypot systems and analysis

I study the impact of deploying active services on the darknet using honeypots as responders. Honeypots have been used in the security activities for years, with well-established projects such as the Honeynet Project [135] and TPot [136] providing several alternatives. Previous works on honeypots have covered many aspects, such as (i) introducing new honeypots that target specific protocols or services [137, 138], (ii) evaluating the effectiveness of different types of honeypots [139], and (iii) presenting techniques to detect honeypots[140, 141]. Readers are invited to review the survey at [12], which provides a broad overview of honeypot research.

Some authors present a general characterization of honeypot traffic, focusing on the origin of attacks, the targeted services, and the frequency of attacks (e.g., [142, 143, 138, 144, 145]). A recent work [146] compared the use of honeypots in different geographic locations. Another work [147] allocated unused addresses in a cloud for honeypot deployment. I revisit these efforts here evaluating the deployment of honeypots compared to what is observed in dark spaces. In addition, I review how measurements from different active responders differ from (and influence) measurements collected in darknets.

My DPIpot leverages DPI to decide how to respond to incoming traffic. This setup allows attacks on non-standard ports to be detected. Some *meta-honeypots* allow flexible configuration of backends to handle traffic on non-standard ports [148, 149, 136]. Most of these systems act as proxies (at various levels) logging the traffic forwarded to the backend. However, they lack DPIpot mechanisms to identify traffic on-the-fly for a variety of protocols.

Honeytrap [150] is the closest honeypot to DPIpot. Honeytrap is a meta-honeypot that performs protocol identification. However, it implements only a limited number of protocol fingerprints. Honeytrap supports about 26 services and provides the ability to extend the set of protocol identification rules. DPIpot instead relies on a state-of-the-art DPI library (nDPI [151]), which is widely used in other network applications and provides hundreds of protocol

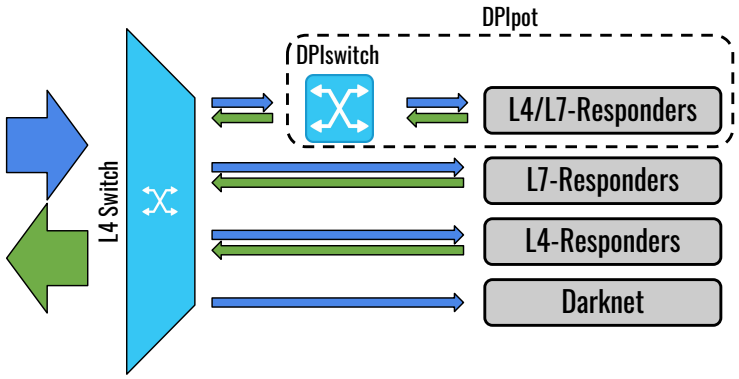


Fig. 6.1 Infrastructure overview. The infrastructure is divided from no interaction (Darknet) to the highest level of interaction (DPIpot).

fingerprints. In fact, I compared different DPI solutions in 5 and concluded that nDPI provides the best coverage and precision for DPIpot.

6.3 Methodology and datasets

6.3.1 Infrastructure

Fig. 6.1 schematically represents my measurement infrastructure. Unsolicited traffic that reaches my dedicated address space is routed – totally unfiltered – to one of my four *deployments* that correspond to different levels of interactivity:

1. Darknet: IP addresses that just receive traffic without responding to any packet;
2. L4-Responders: responders that complete the TCP three-way handshake, capture eventual application requests from clients, but never respond to an application message;
3. L7-Responders: honeypots that mimic popular application layer services. I use state-of-the-art honeypots to simulate well-known services; L7-Responders act as vertical responders that only interact on standard protocol ports, exposing the related emulated service;

4. DPIpot: my novel responder that performs L7 switching of requests using DPI. It decides on-the-fly which protocol to use, and responds to TCP connections on all TCP ports. Unlike L7-Responders, DPIpot decouples the default TCP ports from the application protocols.[2]

Note that none of my deployments respond to UDP or malformed TCP packets to prevent abuse (see ethical concerns in Section 6.3.5).

For the L7-Responders deployments, I rely on the honeypots organized and distributed by the TPot project [136]. 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 a given L7 protocol. Most of my L7-Responders offer login interfaces only [152], 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 [148], or serving pages that mimic actual services accessible over the web [153]. I defer the reader to the documentation of TPot for details.

My 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 on port TCP/3389. To investigate the impact of responding to traffic arriving on other ports, I implement and deploy DPIpot. DPIpot listens on *all* TCP ports. On receiving a new TCP connection request, it completes the three-way-handshake and waits for the first message from the client. Then it analyzes the payload looking for the application-layer protocol. DPIpot relies on `nDPI` [151]. This choice gives us a flexible system that supports hundreds of protocols, which is far more than in previous projects [150]. If a known protocol is found and one of the L7-Responders can handle it, DPIpot directs traffic to such backend; otherwise it acts like L4-Responders. Note that DPIpot can identify and direct traffic only in 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 log in.

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

Both my L4-Responders and DPIpot are implemented in Python using the Twisted framework [154]. My architecture (see Fig. 6.1) is intrinsically distributed, and Twisted is scalable. However, as I will show later, the deployment of responders increases the traffic reaching the darknet by orders of magnitude. To prevent abuses, I thus intentionally limit the capacity of my infrastructure (see Section 6.3.5).

6.3.2 Data capture and processing

I isolate one /23 network to perform experiments with my multiple *deployments* – i.e., darknet, L4-Responders, L7-Responders or DPIpot. My setup is deployed in /16 campus network (at Politecnico di Torino, in Italy) that hosts servers and clients. My infrastructure captures all packets hitting the /23 darknet. I use `tcpdump` and store all traces on a high-end server to generate separate logs for each deployment. [3]

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.

According to the capabilities of each responder, I identify different *flow stages*:

- `SYN`: Flows for which I observe 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;

[3] In this case I forego using high performance tools like α-Mon or DPDK, as I deal with much lower throughputs.

Table 6.1 Deployments and protocols. Each row refers to the traffic of 8 IP addresses during my first capture period (from 15/04/2021 to 16/06/2021). For direct comparison, I report numbers only for the 8 first addresses in Darknet Ext.

Deployment	Category	Addr.	Category: Port	Category: Application	Flows	Flows with L7 Payload	Sender Addr.
DPIpot	*All*	136:143	0:65535	*All below*	1927 M	1207 M	133 k
L7-Responders	*Mail*	128:135	25, 110, 143, 465, 993, 995	pop(s), imap(s), smtp(s)	4 M	71 k	120 k
	Terminal	120:127	22, 2222*, 23, 2323*	ssh, telnet	10 M	5 M	139 k
	Fileserver	112:119	135:139, 445	netbios, CIFS	11 M	6 M	119 k
	Remote Desktop	104:111	3389, 5900, 5901, 5800*, 5801*, 5938*, 6568*	ms rd, vnc, teamviewer, anydesk	13 M	7 M	122 k
	Database	96:103	3306, 33060*, 1433, 4022*, 1434*, 5432*, 27017	mysql, mssql, postgres, mongodb	4 M	212 k	121 k
	Proxy	88:95	8080, 8000*, 3128	generic, squid	5 M	43 k	121 k
	Web	80:87	80, 443	http(s)	4 M	93 k	127 k
	All	72:79	*All above*	*All above*	32 M	23 M	157 k
L4-Responders	*Mail*	64:71	25, 110, 143, 465, 993, 995	-	4 M	36 k	123 k
	Terminal	56:63	22, 2222, 23, 2323	-	6 M	546 k	123 k
	Fileserver	48:55	135:139, 445	-	5 M	1 M	120 k
	Remote Desktop	40:47	3389, 5900, 5901, 5800, 5801, 5938, 6568	-	6 M	546 k	147 k
	Database	32:39	3306, 33060, 1433, 4022, 1434, 5432, 27017	-	4 M	356 k	123 k
	Proxy	24:31	8080, 8000, 3128	-	5 M	38 k	123 k
	Web	16:23	80, 443	-	4 M	59 k	131 k
	All	8:15	0:65535	-	13 M	6 M	146 k
Darknet Int	–	2:5;176:179	0:65535	-	4 M	0	125 k
Darknet Ext	–	2:5;176:179	0:65535	-	4 M	0	111 k

(∗) 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 responds to traffic) and the L4-Responders (which always try to open a connection request).

- `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 for servers to initiate the conversation;
- `L7 payload`: Client and server open the TCP connection and exchange some application-layer messages.

In addition, I 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 the chapter, but included in the public traces I release to the community.

6.3.3 First experiment: Deployments and categories

I perform two experiments. In the first round, I record traffic for two months, from the 15^{th} of April to the 16^{th} of June 2021. This capture starts several months after the deployment of the active responders, thus representing a picture of a stable deployment of active responders in a darknet.

In this first experiment, I split the /23 network into two /24 networks. One /24 network hosts no service and operates as a classic darknet, hereafter called

Darknet Ext – see Tab. 6.1. Unless explicitly mentioned, all results referring to my first experiment and using a darknet as baseline rely upon this *Darknet Ext*.

Then, I split the other /24 addresses in groups of 8 IP addresses. I deploy several categories of responders inside this single /24 as reported in Tab. 6.1 (see the third column). Some host L4-Responders and L7-Responders and respond to 8 specific service *categories*, 8 for L4-Responders and 8 for L7-Responders. Each 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. I create categories for database, file, mail, proxy, remote desktop, terminal, and web services. I report all ports opened for each category on Tab. 6.1, together with some typical applications relying on such ports. I also create an *extra* category denoted as *All*, for which I accept all traffic going to any port. In the case of the *all* category in L4-Responders, I perform TCP handshake for flows arriving in any TCP port. For the L7-Responders category denoted as *All*, I pass 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 8 IP addresses to host DPIpot, which responds in all ports. 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 /24 hosting the active responders act as darknet. I select 8 of these IP addresses and call them *Darknet Int*.

6.3.4 Second experiment: Deploying and removing responders

I perform a second experiment to assess the transient impact of activating and shutting down responders in the darknet. I start by shutting down all active responders on the 25th of January 2022, thus letting the complete /23 to behave like a darknet. This allows us to observe whether (and how) senders continue to search for the responders after they are removed from the network.

Table 6.2 IP allocation in the fresh deployment of active responders.

Deployment	**Category**	**Addr.**
Darknet_4	–	192:255
DPIPot	*All*	184:191
Darknet_3	–	128:183
L7-Responder	*All*	120:127
Darknet_2	–	64:119
L4-Responder	*All*	56:63
Darknet_1	–	0:55
Darknet^*_{ext}	–	–

(∗) **Darknet**$_{ext}$ in this experiment is equivalent to **Darknet Int** in Tab. 6.1.

On the 9th of February 2022, I light up fresh responders in the /24 that previously served as darknet (*Darknet Ext*). This /24 had been used as darknet for many years before the start of my experiments. As such, it allows us to observe the speed senders discover new services as well as all transition steps from a darknet IP address into an active responders.

I deploy L4-Responders, L7-Responders, and DPIpot using 8 IP addresses for each deployment, which are distributed in the /24 network as reported in Tab. 6.2. In this experiment I use only the *All* category for L4-Responders and L7-Responders. The deployment is instrumental to maintain an equally spaced set of dark IP addresses between each group of active responders. This would let us measure whether the placement of active responders impacts the neighboring addresses.

6.3.5 Ethics

I take several countermeasures to restrict the impact of my measurements on third-party networks. First, and most important, I never send packets if my packets may worsen the position of attack victims. In particular, I never send UDP traffic, as it could make my infrastructure part of DDoS attacks relying on spoofed addresses and amplification techniques. For the same reason, I silently drop all TCP packets with `SYN/ACK` flags and other malformed flows, as they

may arrive from victims of DDoS attacks with spoofed addresses. Responding to such packets may help the attackers to overload the victims' networks.

The traffic I observe may come from infected machines that are taking part in botnets. As previously said, I explicitly limit the capacity of my infrastructure to avoid creating too much traffic for the networks hosting such infected machines. The setup discussed in this chapter 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 my infrastructure may uncover vulnerable computers exploited by attackers [155]. I take all measures to protect such IP addresses. I anonymize addresses in the datasets I release publicly. I also collaborate with my security team and my upstream providers, actively notifying about novel attacks and senders.

6.4 Macroscopic changes in traffic

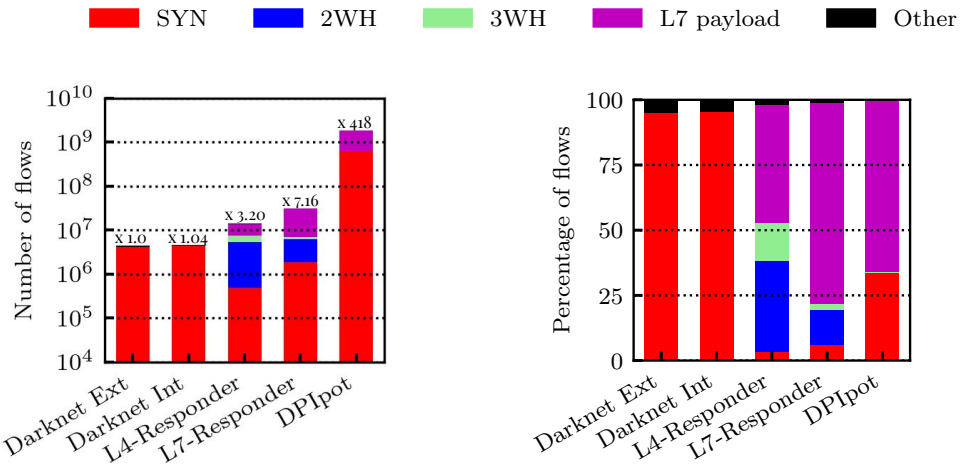
I report a high-level characterization of the different deployments aiming to answer the following question: How much extra information one would get when some IP addresses inside a darknet actively respond to incoming traffic?

For easy comparisons, I restrict my analysis to 8 addresses per deployment. I focus on those addresses in the *all* category in the case of L4-Responders and L7-Responders, and get 8 addresses from *Darknet Ext* and *Darknet Int* (see Tab. 6.1). Here I focus on my first experiment setup, and whenever not explicitly mentioned I report statistics for the first month of my dataset to easy visualization.

6.4.1 Breakdown per flow stage

Fig. 6.2 reports the number of 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.

Darknets observe a large majority of TCP `SYN` messages, with a few UDP and bogus TCP segments (about 8% of the total). As soon as I start replying



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

(b) Traffic share in the deployment

Fig. 6.2 Flows reaching different deployments. Numbers inside the left plot mark the increase in respect to *Darknet Ext*.

with the L4-Responders, the number of flows grows by a factor of 4 compared to the darknets[4] (cfr. Tab. 6.1). Although my deployment shall perform the full 3-way handshake, a small portion of flows remain in the `SYN` stage, i.e., connection requests to which my L4-Responders deployment cannot reply due to short-term congestion. Interestingly, 35% of the flows terminate at the `2WH` stage, most likely corresponding to "TCP-SYN scans" (also reported in [122]). About one forth of the open TCP flows carries no payload, i.e., likely host discovery actions performed with a "TCP-connect scan".

Consider now the L7-Responders. the number of flows doubles again. The `SYN` stage flows are now about 7%. Part of this traffic is again caused by the limits I impose on my infrastructure. However, as I will see later, once I respond to 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 72% of the total.

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

[4]Other subsets of darknet addresses yield a stable number of flows

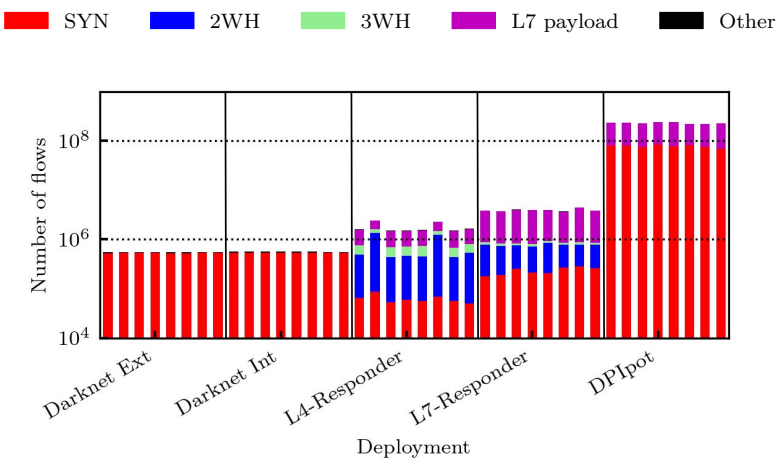


Fig. 6.3 Number of flows per deployment. Each bar in the figure reports numbers for a single IP address of the several deployments.

more than in the darknets. Here I see around 40% of cases finishing on `SYN` stage, which correspond to periods in which my deployment hits it capacity.

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

Fig. 6.3 reports the number of flows observed for each IP address selected for this analysis. Here, I see that the number of flows reaching each deployment is well-divided among the IP addresses belonging to the given deployment. Little variations are seen for the L4-Responders, where a couple of sources contribute with large-scale attempts against two L4-Responders IP addresses.

More interesting, the number of unique sources contacting each deployment changes considerably (see numbers in the last column of Tab. 6.1). Differences are visible between *Darknet Ext* and *Darknet Int*. In fact, IP addresses belonging to *Darknet Int* attract thousands of sources more than those in *Darknet Ext*. I conjecture that this behavior is a consequence of the presence of active responders in the same /24 subnet. Once sources find services in a subnet, they search for services in the neighbour addresses. I will investigate this in more details in Section 6.7.

Finally, L4-Responders, L7-Responders and DPIpot attract more senders than the darknets. In sum, deploying active responders sheds light on new scanners and attackers that would not be uncovered with simple darknets.

6.4.2 Temporal evolution

Darknets and honeypots are known to receive variable traffic over time. Fig. 6.4 reports the average per-hour number of flows received by each deployment (*All* category). Here I report time-series covering the full 2-month dataset of my first experimental setup. Notice 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 scans hitting the address space [113, 119, 121]. Both L4-Responders and L7-Responders show a noisier pattern over time, again with small episodes of increases. The 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 Section 6.6.3. As said above, these episodes bring DPIpot to the limits I impose on the infrastructure.

Takeaway: The number of flows grows by orders of magnitude with increasingly interactive responders. Vertical honeypots attract many times more flows than darknets. DPIpot pushes this increase further thanks to its ability to respond to application traffic on non-standard ports. This growth creates temporal bursts of traffic that challenge the deployment itself and calls for protection mechanisms to avoid collapsing the infrastructure and biasing the collected data.

6.5 Ports and senders

I have observed the effects of answering darknet traffic in terms of traffic volume. I now assess changes on traffic patterns along two axes: the targeted services and sender IP addresses. This section answers the following question: How the presence of active services changes the behaviour of the groups of senders and of contacted services?

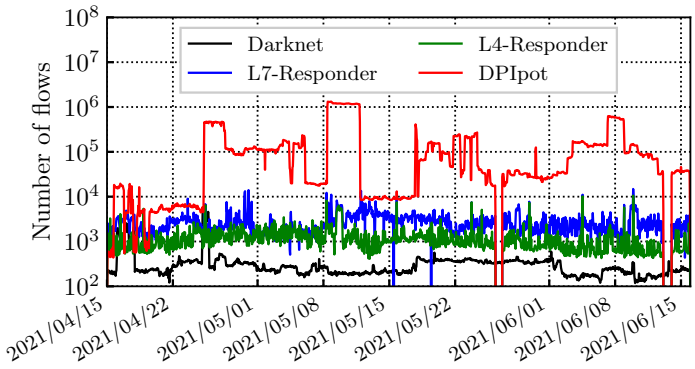


Fig. 6.4 Temporal evolution of the number of flows.

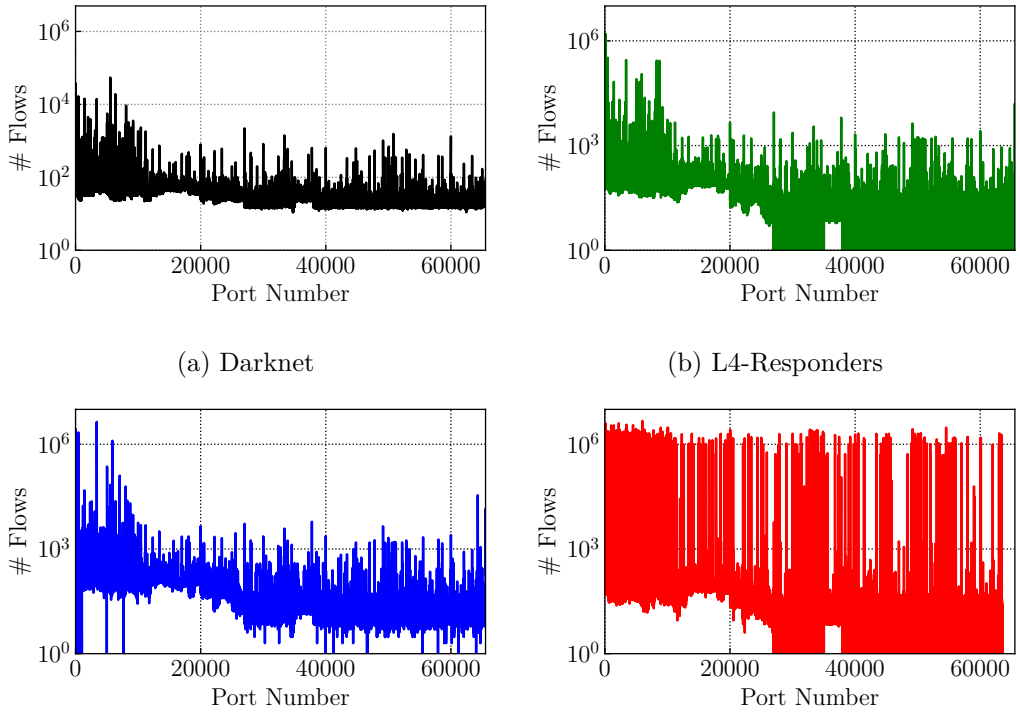
For this analysis I again rely on my first experimental setup, i.e., long-run assignment of address to active responders.

6.5.1 Changes on probed ports

The traffic volume is expected to vary over the exposed ports. Already in a darknet, well-known ports are expected to be more frequently contacted than others. Fig. 6.5 reports the number of flow per port for the different deployments. Here I see interesting effects of deploying the active responders. Common for darknet, L4-Responders and L7-Responders, senders concentrate their interest on well-known ports below 1024. On the contrary, DPIpot attracts much more flows on very uncommon ports (notice the y-log scale). Investigating the `L7 payload`, these flows are related to Remote Desktop Protocol (RDP), hinting for a specific attack (I investigate this in Section 6.6).

In detail, focus on the darknet (black plot on the left). While some ports do receive a much larger share of traffic as expected, scanners cover the whole port range. This confirms how darknets are effective to observe senders performing horizontal scans, i.e., doing host-discovery.

When I start responding to requests, the picture drastically changes. For instance, the top ports in the L4-Responders account for more than 60% of the flows. This percentage grows to more than 70% in L7-Responders. See how the number of flows increases for low, well-known ports in Fig. 6.5-b and Fig. 6.5-c. That is, once a target is discovered, senders activate the next stages



(a) Darknet

(b) L4-Responders

(c) L7-Responders

(d) DPIpot

Fig. 6.5 Number of flows per destination port for the four deployments. The presence of different active responders changes the observed traffic per port.

of scans or attacks. Interesting, senders contacting L4-Responders skip some ports starting from port 27 000 (the 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.

I observe a completely different picture for DPIpot. Some hundreds of ports get millions of flows, and the remaining ports gets some uneven distribution of traffic. Unlike the darknets and L7-Responders, more than 15 000 ports never received any flows, similarly to the L4-Responders case. Recall that, for both L4-Responders and DPIpot cases, all ports result *open* during port scans. I conjecture that either senders get trapped performing activities on the found open ports, or they are more cautious and abort (or time out) scans after finding a high number of open ports.

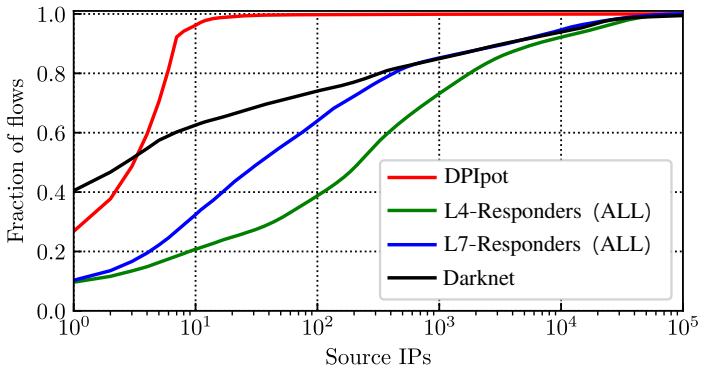


Fig. 6.6 Fraction of flows per sender IP address.

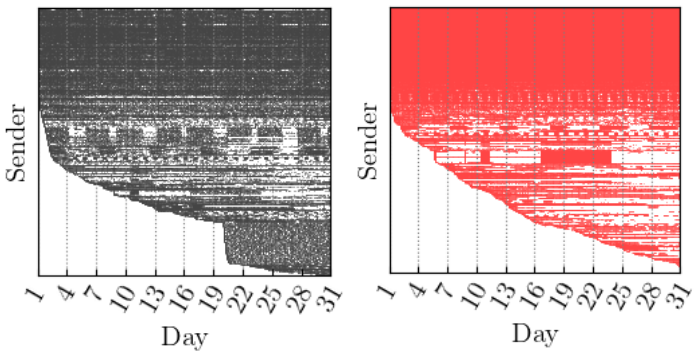
Takeaway: Active responders in some ports engage senders, which activate the next stage in their scans or attacks. This increases the traffic and sometimes challenges the monitoring infrastructure. Enabling all ports traps senders in some activities, possibly limiting/biasing their activity.

6.5.2 Changes on traffic senders

We now investigate changes seen in the set of senders contacting the deployments. We start by highlighting the last column of Tab. 6.1. The number of unique senders varies substantially across the deployments. In fact, senders increase by around 40% in the L7-Responders when compared to the *Darknet Ext.* Even more interesting *Darknet Int*, i.e., those IP addresses hosted in the same subnet with the responders, observe around 12% more senders than the pure darknet subnet.

We dig into the behavior of these senders in Fig. 6.6. It reports the cumulative fraction of flows from each sender. The x-axis reports (in log scale) the rank of sender IP addresses according to the volume for each deployment.

In the darknet, the three most active senders generate 40% of flows. These are well-known scanners reported multiple times in blocklists. The top-10 most active senders are responsible for 63% of the flows. Some of them are always active. Some senders probe at high rate (hitting 20 000 flows per hour) and disappear. We also observe a tail of more than 63 000 IP addresses. This tail is inline with previous work [113] that shows darknet traffic is dominated by



(a) Darknet (b) DPIpot

Fig. 6.7 Activity pattern of top-1000 sender IP addresses. Each row corresponds to a sender IP address.

bugs and misconfiguration, with only a minority of senders actually performing scans and attacks. Flows are more distributed across senders in L4-Responders and L7-Responders, with the top-10 accounting for 20% and 32% of traffic, respectively.

The figure is completely different in DPIpot where the top-10 most active senders account for 95% of the flows. These senders are involved in RDP abuses observed on non-standard ports. They generate millions of flows per hour, triggering our rate limiting countermeasures.

Fig. 6.7 offers a visual representation of the activity of the top-1000 most active senders over time. Each row corresponds to an IP address. A dot is present if that IP address is active in that hour. We 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 in the day 20 in the darknet. These senders are likely bots that perform a coordinated scan reaching our address space. In general, we can distinguish different patterns: some senders are persistent, while others keep coming back periodically. A few senders interact only for some time before disappearing and never coming back. The latter is more visible in Fig. 6.7b for DPIpot.

We complement the analysis in Fig. 6.8, where we investigate scan patterns performed by the top-100 most active senders. We show only darknet for brevity. The x-axis reports the destination port of flows, sorted by value. Each

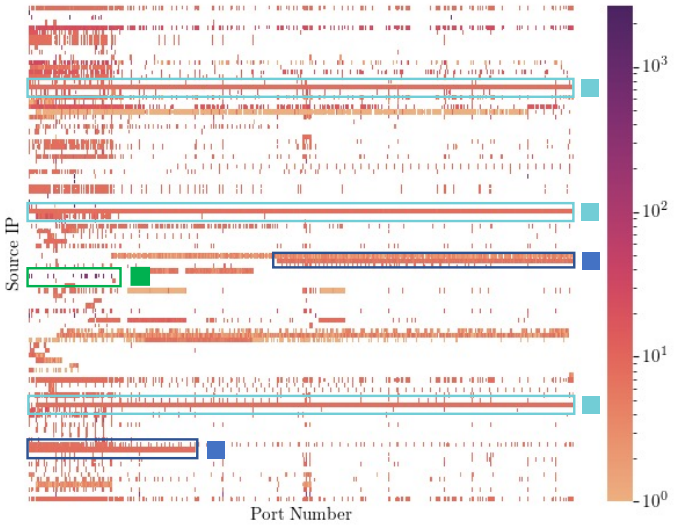


Fig. 6.8 Top-100 senders vs. destination port. Addresses are sorted numerically.

row refers to a single sender IP address. A dot is present if that IP address sends a flow to such port. The darker the color, the larger the number of flows.

For readability, we highlight some patterns with colors. First, some few horizontal scanners are visible (cyan). These senders check all ports, even in the darknet. Second, we observe some vertical scanners (green) - i.e., senders that sends lots of packets for few ports. Third, some senders cover a large set of continuous ports, covering a subset of all ports (dark blue). All these patterns are seen for other deployments too, which however show yet other behaviors. For example, besides the horizontal scanners, DPIpot allows us to observe very targeted scans on few ports, i.e, vertical attacks, and a large numbers of coordinated scanners, i.e., groups of senders that target the same few ports simultaneously. This has been confirmed in [156]: it leverages embeddings to discover common sender patterns.

Takeaway: The presence of active responders attracts a new wave of senders, which target also the addresses remaining dark in the subnet. Most of traffic comes from few thousand senders that are involved either with vertical or horizontal scans and attacks. Unlike vertical honeypots, DPIpot lets us observe scan patterns where also non-standard ports get the attention of attackers.

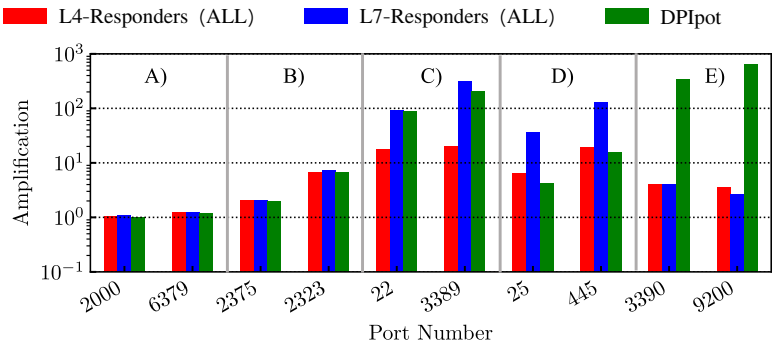


Fig. 6.9 Amplification factor for the most targeted ports.

6.6 Amplification of service-specific deployments

I now focus on the extra information different responders offer compared to darknets I consider my first experiment.

Here, I answer the questions: Does the presence of specific services attract traffic to other services? What happens when one deploys services on non-standard ports?

6.6.1 Service amplification

To quantify the extra traffic per deployment, I define the *amplification factor* as the ratio between the number of flows seen on a given port(s) for the 8 IP addresses of a specific deployment, and the number of flows directed to the same port(s) on the 8 IP addresses belonging to the *Darknet Ext.*

First, I run a preliminary test to verify whether the amplification factor changes when comparing IP addresses belonging to the same deployment. For this, I take all groups of 8 sequential IP addresses (/29 subnets) in the *Darknet Ext* and compute – for each destination port – the amplification factor for each group pair. Not reported here for brevity – the distribution of the amplification factors is centered between 0.9 and 1.1. I therefore consider significant any amplification factor outside this range.

Fig. 6.9 shows the amplification factor for some selected ports. I identify five major behaviors, which I label with capital letters and for which 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 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 amplification factors. L4-Responders are less interesting for the senders, with reduced amplification factor;

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 services on non-standard ports and perform attacks. Only DPIpot, the only one able to identify the L7-protocol, let us quantify 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 amplification grows to almost 1,000.

Takeaway: Different deployments amplify different behaviors. Overall, the obtained information clearly grows from case A) to case E). The latter is

Table 6.3 Amplification for L4-Responders and L7-Responders. Cases in which no amplification is observed are marked with a hyphen.

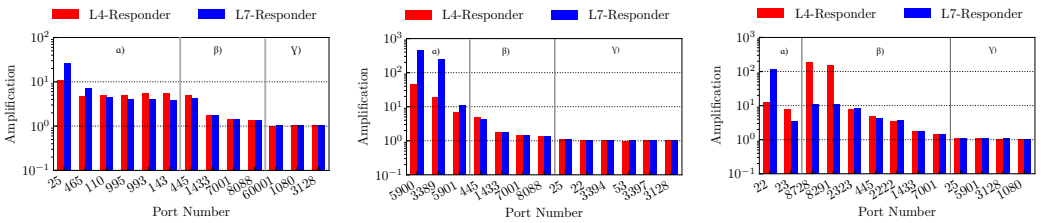
L4-Responders								
	DB	*File*	*Mail*	*Proxy*	*RD*	*Terminal*	*Web*	*Others*
DB	**15.4**	4.3	–	–	–	–	–	–
File	1.6	**42.0**	–	–	–	–	–	–
Mail	1.5	4.1	**6.5**	–	–	–	–	–
Proxy	1.5	4.2	–	**2.7**	–	1.2	–	1.2
RD	1.5	4.2	–	–	**21.2**	1.6	–	–
Terminal	1.5	4.1	–	–	–	**9.3**	–	1.3
Web	1.5	4.2	1.4	1.2	–	1.3	**8.1**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

particularly interesting, showing effects of performing traffic analysis on non-standard ports. DPIpot leads to the discovery of active attacks on non-standard ports, otherwise unseen with darknets or L4-Responders. A simple darknet would offer a much limited view overall.

6.6.2 Targeted services and *Side-Scans*

So far I evaluated 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. I consider L4-Responders and L7-Responders, with categories/ports/applications defined in Tab. 6.1. For each category, I compute the amplification factor with respect to the corresponding categories/ports/applications in darknet addresses.

Tab. 6.3 summarizes results. For each vertical deployment, I report the amplification factor only when significant. Rows report the category of the deployment while columns report the corresponding categories in the *Darknet Ext* 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 amplification factor than L7-Responders in some cases (e.g., *DB*). This is likely a consequence of my lack of honeypots



(a) Mail (b) Remote Desktop (c) Terminal

Fig. 6.10 Amplification factor for selected deployments. β marks cases of *Side-Scans*.

in the L7 backends (see Tab. 6.1). In this case, while L7-Responders reply with an uninteresting response (reset the connections), the limitation to the TCP handshake offered by L4-Responders further engages the senders.

I observe also significant amplification factors on services for which the deployment does *not* answer, i.e., where I drop the `SYN` packets. Regardless of the deployment, once senders find an IP address that is alive (i.e., hosting 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. 6.10 reports some of the most relevant *Side-Scans*.

α) marks the well-known (and open) ports for the category. Here as expected I get significant amplification factors, 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. For instance, when opening ports of the *Mail* category (plot in the left-hand side), I observe significant amplification factors on ports (445, 1 433), which are usually used in *File* and *DB* services. I see curious *Side-Scans* also on ports (7 001,8 088). Similar effects can be seen for the *Remote Desktop* category.

More expected, ports (2 222, 2 323) are often used as alternative ports for terminal services – and senders *Side-Scan* these ports when finding standard

Table 6.4 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

terminal ports open (right most plot). Ports (8 728, 8 291) are known to be vulnerable services in old versions of software routers. I observe frequent "door-knocking" attempts: the sender checks port 22 first; if open but no banner is offered, they check 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 [157].

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.

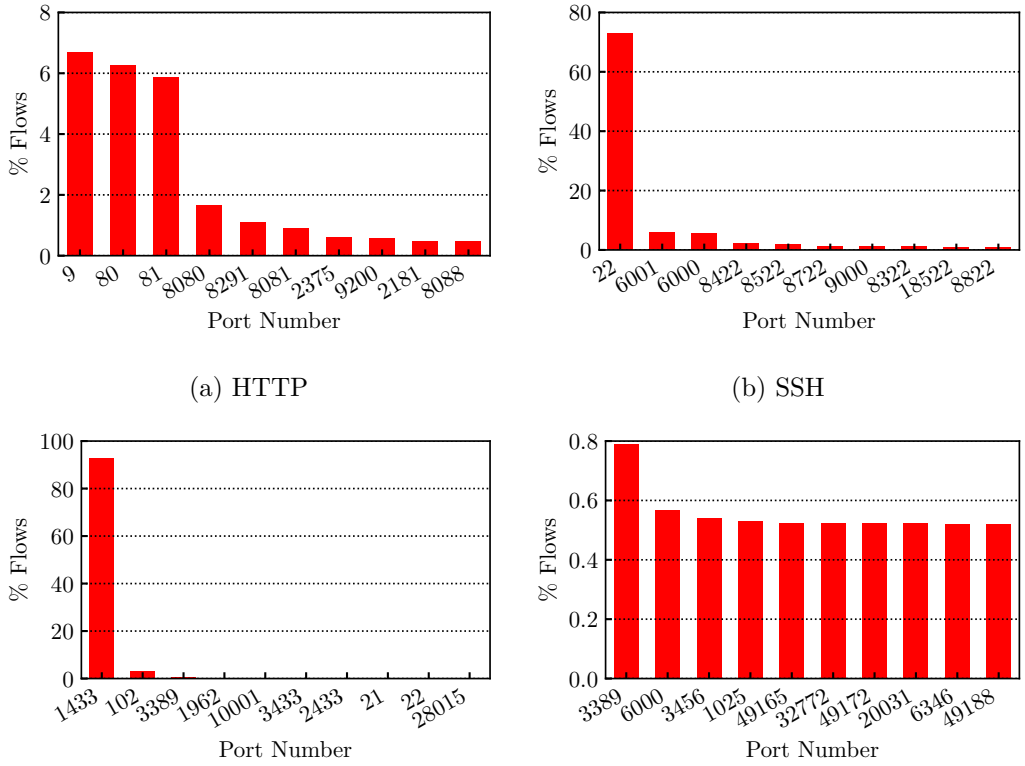
Takeaway: I observe high amplification in both L4-Responders and L7-Responders. Deployments targeting a particular service uncover *Side-Scans*, which vary according to the service exposed and the behavior of the responder.

6.6.3 DPIpot additional visibility

We now dig into DPIpot data to check the *Side-Scan* phenomena in this case. Tab. 6.4 shows that DPIpot observed 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. This behaviour is also seen in Fig. 6.5 and Fig. 6.6 where the IP addresses involved in this attack dominate the traffic DPIpot collects.

DPIpot lets us observe also other popular protocols like HTTP, TLS and SSH, with multiple senders targeting 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 are to the default ports.

To check how senders choose the port to probe for a given protocol, Fig. 6.11 details the most popular target ports for some L7 protocols. Start from the



(a) HTTP

(b) SSH

(c) MsSQL-TDS

(d) RDP

Fig. 6.11 Flows percentages on top-10 ports for DPIpot and different L7 protocols.

HTTP case. Port 9 results as the most popular port. 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 [158]. This scan activity would likely go unnoticed on traditional honeypots. Besides this curious scan, DPIpot recognizes HTTP requests on non-standard ports that it correctly handles. 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 check other ports where system administrators may move the SSH service, e.g., 8 422, 8 522, 18 522. This behavior suggests a targeted *Side-Scan* where senders generate the port to target with some domain-driven algorithm. The *Side-Scan* using the MsSQL-TDS protocol is even more vertical. Most of the attacks are

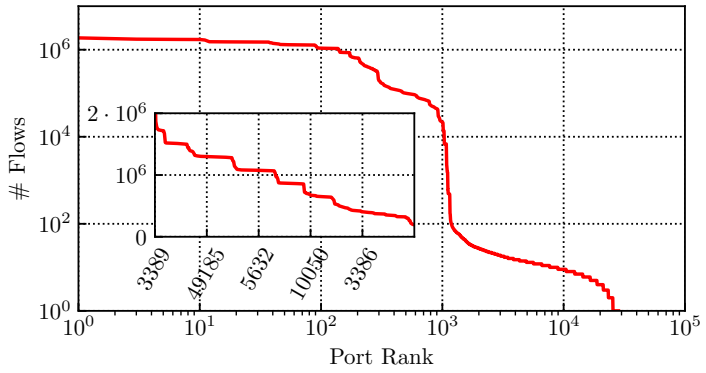


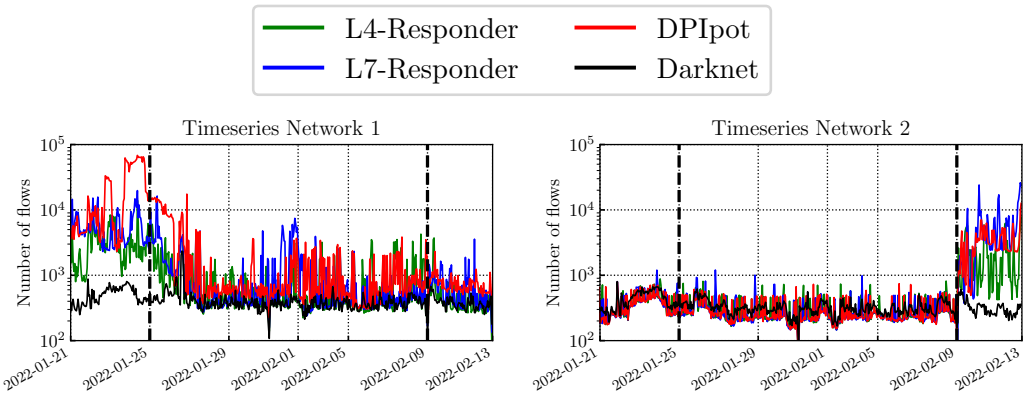
Fig. 6.12 Flows per port (RDP). Zoom on first 300 ports in inner axis.

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 [159] via attacks that start with password brute-force [160] as well as a common backdoor [161].

Thanks to DPIpot, we observe 1 415 senders performing password brute-force attacks. The attackers however execute the brute-force in almost any port announcing RDP support. Fig. 6.12 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 receives 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 attack. The inner plot shows that there is also a clear pattern for the top-300 ports. Checking which ports each sender targets, we 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



(a) Powering off responders (IP ranges in Tab. 6.1)

(b) Powering up responders (IP ranges in Tab. 6.2)

Fig. 6.13 Effects of removing and adding active responders in darknets.

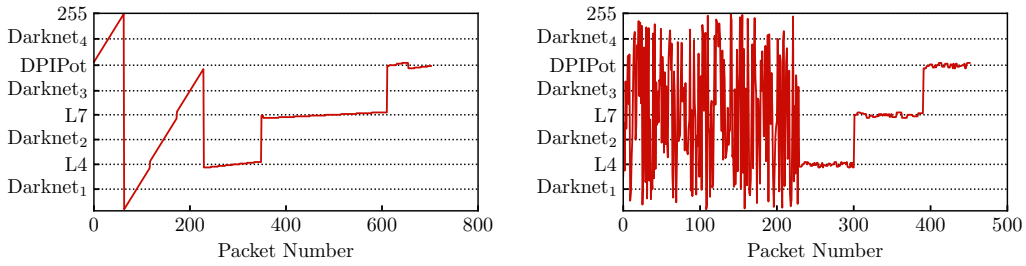
- Senders that scan thousands of ports (16 IP addresses). These addresses have been reported as heavy scanners [162] and perform very similar activity. This suggests they are part of the same botnet.

Takeaway: DPIpot unveils unexpected *Side-Scan* attacks and scans where senders target non-standard ports. It also triggers activity that L7-Responders in the standard ports do not observe. Senders may become very aggressive, calling for precaution to avoid overloading the monitoring infrastructure.

6.7 Darkening and enlightening networks

I now shift my attention to my second experimental setup, in which I shut all active responders down, before enlightening new active responders in the other darknet.

I answer the following questions: Do senders continue to reach IP addresses that once hosted active responders? How fast does a newly-active IP address become target of the senders unseen in the darknet? How does the deployment of active responders impact neighboring IP address?



(a) Sequential host scan - targeted port scan on responders

(b) Random host scan - targeted port scan on responders

Fig. 6.14 Example of host discovery patterns observed in the infrastructure.

6.7.1 From light to darkness, and back

Fig. 6.13 describes the traffic evolution for some deployments in my infrastructure. Let us focus first on the deployments that have been shut down. Fig. 6.13a depicts time-series of the number of flows per hour for groups of 8 IP addresses hosting the *Darknet Ext*, L4-Responders (All), L7-Responders (All) and DPIpot in my first experimental setup (see Tab. 6.1). Before the shutdown (first black dashed vertical line) the active responders observ more than 10^3 flows per hour, whereas *Darknet Ext* between 10^2 and 10^3 flows per hour, respectively. The number of flows per hour remains orders of magnitude higher in the active responders when compared to *Darknet Ext* even two days after the responders are down. In fact, the traffic remains noisier for the IP addresses that were hosting the responders for weeks. In sum, the senders that target the active responders insist on reaching these responders, and the traffic does not return to the darknet levels even two weeks after the shutdown.

Focus now on Fig. 6.13b which depicts the deployment of fresh responders in the network that originally hosted *Darknet Ext*. Before the activation of any service, all groups of IP addresses observe the same amount of traffic (10^2–10^3 flows per hour). As soon as I deploy active responders on Feb 9th 2022, I spot an immediate increase in traffic for all cases. I will show later that this increase is partly caused by a new wave of senders that immediately and suddenly reach each responders to perform an in-depth port scan. This result hints for coordination with those senders that perform initial host discovery.

Again, it confirms the advantage of having a deployment that mix both types of responders.

To shed more light on senders' strategies for port and service scanning, Fig. 6.14 shows two examples of common patterns observed when the responders are deployed. The figure depicts the sequence of IP addresses a given sender targets over time. On the y-axis, I report the type of darknet/responders on such IP addresses.

The first example of Fig. 6.14a illustrates the behavior of sequential scanners. These scanners sequentially visit every IP address in the /24 subnet to find open services. After this host discovery, they get back to those IP addresses hosting responders to perform in-depth port scans and application attacks. Some scanners start from a random initial IP address (as in Fig. 6.14a), while others start from the first address in the /24 subnet.

Fig. 6.14b instead shows an example of a scanner that performs a random host scan: these scanners keep contacting random IP addresses in the /24 subnet to perform host discovery. Once this stage is completed, they come back to those responders for in-depth activity, similarly to the sequential scanners.

Takeaway: It takes days or even weeks for senders to stop targeting responders that went offline. Conversely, as soon as an IP address is found responding to traffic, it becomes target of (new) senders almost immediately to perform an in-depth host and service discovery. Senders employ diverse strategies to perform the discover activities.

6.7.2 Disturbing the neighbours

I further investigate if the presence of active responders causes disturbance to IP addresses remaining dark in the same /24 subnet. This question is important for those running darknets to understand to what extent active responders pollute the darknet traffic. Recall from previous sections that while active responders do attract more senders, they sometimes trap senders in specific activities, thus biasing senders' behaviors. Here I verify how neighbor addresses are impacted.

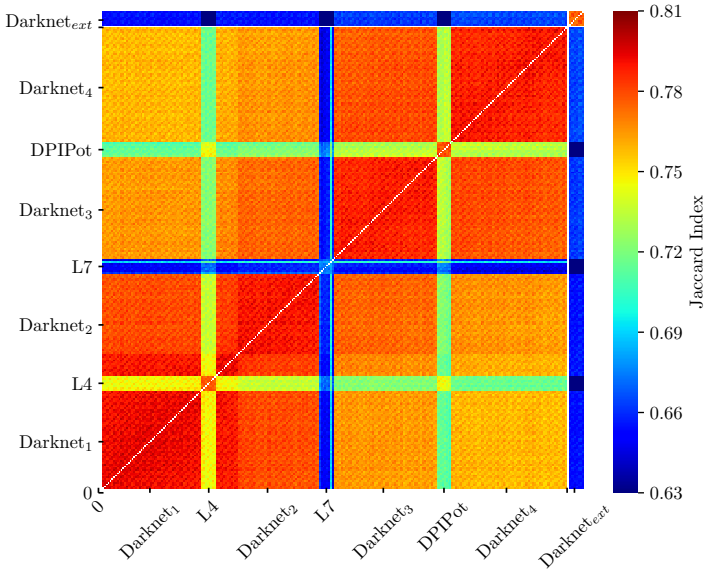


Fig. 6.15 Jaccard Index among aggressive senders targeting each IP address.

Recall from the last column of Tab. 6.1 that the number of senders varies substantially across the deployments. I confirm such behavior in my second experiment. For those darknet IP addresses close to active responders, the increase in the number of senders starts immediately after the responders become active.

I now investigate if the additional senders contacting dark addresses are similar to the ones reaching the responders. For this, I compute the Jaccard Index for all pairs of addresses in the /24 subnet where I have deployed fresh responders. To filter out occasional senders, I restrict the analysis to *aggressive senders* – those sources that send at least 100 packets over 1 month.

Fig. 6.15 shows the Jaccard Index in the form of a heatmap. Overall, the figure shows two main effects: 1) active responders attract a different set of senders, 2) there is a pollution effect, but not directly nearby the responders.

For 1), notice the low Jaccard Index when comparing active responders with darknet addresses (e.g., the rows/columns corresponding to L4, L7 and DPIpot). This decrease is due to an increase in the number of senders that target *only* the responders (causing an increase in the denominator of the Jaccard Index). This behavior confirms that some senders perform only the “host scan” phase,

while other senders become active to perform subsequent phases of attacks, e.g., “port scans”, “application scan” and ”vulnerability exploitation” on addresses that are found alive.

For 2), darknet addresses at the beginning (end) of the /24 address space tend to observe a higher fraction of senders in common with neighboring addresses (causing an increase in the numerator of the Jaccard Index). This fact is reflected into the darker red pattern seen along the diagonal of the Jaccard Index. This is an effect of the sequential scanners that stop their activity before complete the scan of the entire /24 subnet.

Finally, focusing on the Jaccard Index computed among addresses in the external /24 darknet (top and rightmost groups), I observe a different set of senders. This behavior is due to the set of senders scanning one /24 to be different from the set of senders scanning the second /24.

Takeaway: Senders involved in darknet scans are typically different from those seen in subsequent attack stages. These new senders are seen only when active responders are present. Interestingly, the present of responders attract new senders also for addresses remaining dark.

6.8 Conclusion

I systematically analyzed the impact of deploying interactive responders on the darknet address space. My results show the clear benefit of engaging with senders, with more and more interactive responders that allow one to collect richer data on the senders' behaviors. I also showed that a careful design in the deployment, with the ability to turn on and off responders at need, offers even more opportunities, uncovering a new wave of senders that otherwise would remain unobserved.

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 my 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. Distributing my active responders to other IPv4 ranges, IPv6 networks and different geographical locations is also a challenge that should be faced in future work.

Chapter 7

Exploring Application Level Attackers' Interactions

After having explored the activity in the network at Network/Transport Layer (L3/4), in this chapter I focus my attention on what happens at the Application Layer (L7) by collecting and analyzing Honeypots logs. Here I present my findings discussed in *What Scanners do at L7? Exploring Horizontal Honeypots for Security Monitoring* presented in *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)* [163].

7.1 Introduction

To gain visibility into network attacks, *honeypots* are a common means to collect data. Usually, honeypots are deployed as vertical systems targeting a specific scenario [12] such as databases, terminal servers, or web applications [164, 148, 153], each supporting the respective protocols.

Researchers have focused on different aspects when dealing with honeypots. Most work has focused on the introduction of new honeypots, focusing on attacks against particular services [137, 138]. Others evaluate the effectiveness of different honeypots [139] in exposing useful data. Another body of works focuses on techniques to detect, and avoid the detection, of honeypots [140, 141]. Considering the analysis of honeypot logs, previous works compare deployments in different geographic locations [146] and study how attackers respond to

different honeypot sophistication [165]. However, such works focus on single deployments or services, providing an in-depth analysis of logs in these vertical deployments.

7.1.1 My Contribution

I here aim at extending these analyses by revisiting the visibility offered by honeypots from a *horizontal perspective*. I compare if and how the same attackers interact with different honeypots offering multiple applications. Do attackers typically attack a single system or do they extend the attack surface on multiple systems? Do they use the same strategies for multiple honeypots?

To answer these questions, I rely on the infrastructure depicted in Ch. 6. It provides multiple low-interaction honeypots. I collect data for 5 months at the transport and application layers, recording millions of application requests from tens of thousands of sources. Exploiting this perspective, I observe scanners that always attack the same service as well as horizontal attackers targeting multiple services simultaneously. In this chapter I'll investigate at application level the what has been discussed in Ch. 6.7.1 to observe attackers that dismiss the previous targets, discover new systems and return to full-speed campaigns.

Finally, I dig further into the dataset, providing an initial horizontal analysis of the brute-force attacks against multiple services. I study the credentials used on login attempts against my honeypots, collecting passwords used on each system. I observe attackers using password found in known data breaches [166], but also groups of attackers relying on other password lists.

I revisit and update known facts about honeypot deployments, highlighting the greediness of some attackers against multiple services.

I believe this work offers some insights that highlight the benefits of a horizontal perspective when characterising attacks captured by honeypots. The complexity and multi-facet nature of the data honeypots expose calls for cooperation on the deployment of (open) honeypot infrastructure and on the sharing of honeypot data. For that, I here report initial analysis of my dataset and deployment, which are available to other researchers upon NDA agreement to protect eventual sensitive information present in the data.

After describing my deployment in Section 7.3, I provide a general characterisation of L7 traffic reaching my deployment in Section 7.4. Then, I describe the brute-force attempts recorded by multiple honeypots in Section 7.5. Section 7.2 summarises related work, while I list conclusions and future work in Section 7.6.

7.2 Related Work

Honeypots have been deployed for cybersecurity purposes for a long time and multiple honeypot projects exist, as already discussed in Ch. 6.2.

Some authors [138, 144, 145, 167] present general characterisation of honeypot traffic, focusing on the origin of attacks, targeted services, frequency of attacks etc. Most previous work is however focused on vertical deployments, looking at the activity recorded in honeypots deployed for a particular type of attack, eventually deployed in several regions and networks [168, 169].

I instead report initial data captured with multiple honeypots simultaneously. I shed light on the differences and similarities of the traffic observed in this heterogeneous setup, reporting not only patterns in terms of traffic sources, but also common activities for multiple L7 services, such as brute-force attacks across various honeypots [170, 171].

7.3 Methodology and Dataset

I set up an infrastructure that relies on the honeypots organised and distributed by the T-Pot project [136]. In this work I report on a subset of the T-Pot honeypots, excluding Telnet and SSH, which I leave for future work. All honeypot services are low-interaction honeypots [12]. I configure T-Pot to expose the services listed in Table 7.1. Each honeypot logs and registers all application (L7) interactions, including brute-force login attempts, requests to specific service functionality etc.

Recall from Ch. 6.3.1 that I use two /24 networks of a university network that I reserve to run my experiments. The honeypots are thus hosted in a regular campus network, with no firewall to protect them. To monitor all the

Table 7.1 Honeypot services and amount of traffic seen in my deployment.

Service	Sender Addr.	L7 Requests
smb [164]	30 854	25 348 188
http [164, 153]	18 370	1 931 553
mssql [164]	7 119	396 391
rdp [172]	4 813	46 036 456
ftp [164]	1 845	20 834
mysql [164]	1 527	32 091
mongodb [164]	1 364	21 112
epmapper [164]	1 203	71 014
pptp [164]	912	59 488
smtp [173]	827	3 994 871
mqtt [164]	567	4 287
echo [164]	314	20 378
vnc [152]	172	12 741 040
postgresql [152]	27	28 163
pop3 [152]	12	26 194

incoming traffic, I record packet-level traces using tcpdump, which are regularly processed along with all logs of the honeypots.

I activated my deployment on October 27th, 2021. I here analyse data collected until March 1st, 2022. Specifically, from October 27th, 2021 to January 25th, 2022, I configured my honeypots as depicted previously in Tab. 6.1 and considered just the 8 IP addresses in the first /24 network related to L7-Responders. All IP addresses expose precisely the same services (see Table 7.1). On January 25th, 2022, I shut the honeypots down for two weeks, after which I restarted all services on February 9th, 2022, using 8 previously silent IP addresses in the second /24 network. I perform this experiment to observe patterns related to the discovery and subsequent attacks against new systems.

I collected about 100 *million* application layer (L7) requests coming from more than 57 000 unique IP addresses. In the following, I generically refer to these IP addresses as "attackers", i.e., senders that have interacted with one of my honeypots at application layer at least once during the observed period. In this work I thus ignore cases where a real attacker may use multiple IP addresses, or cases where multiple real attackers reach my systems from the same IP address, e.g., on different time periods. Notice that not all of such senders are malicious. I will show later that many senders are actually crawlers, e.g., from security companies. I will highlight these cases, and give particular attention to the clearly malicious activity, such as brute-force password attacks.

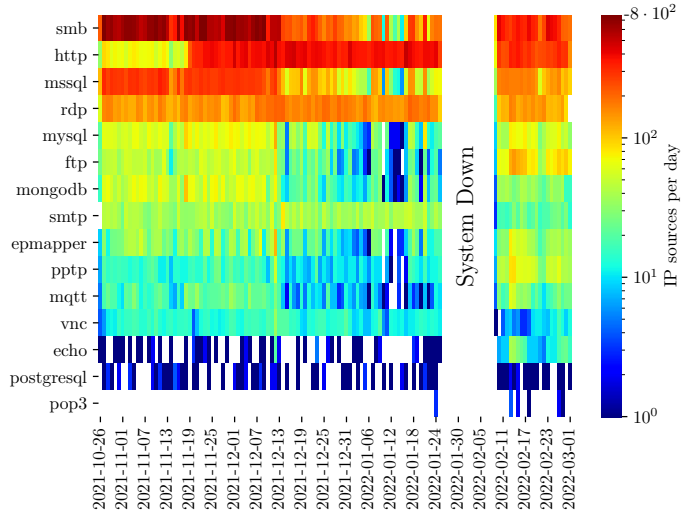


Fig. 7.1 Number of IP sources per service for each day of observation.

Table 7.2 Distribution of traffic per service coming from some known sources.

Class	Service (% L7 Requests)									
	smb	http	rdp	ftp	vnc	smtp	mongodb	mqtt	mysql	others
Mirai	16.13	1.86	**59.44**	0.1	16.98	4.78	0.02	-	0.03	0.66
Google	-	**100**	-	-	-	-	-	-	-	-
Shadowserver	14.52	18.55	**37.10**	12.10	-	-	13.70	4.03	-	-
Shodan	-	-	12.66	22.15	15.82	**26.58**	-	12.03	10.76	-
Onyphe	**100**	-	-	-	-	-	-	-	-	-
Stretchoid	-	**100**	-	-	-	-	-	-	-	-
Unknown	**97.94**	0.45	0.89	0.17	-	-	0.28	0.17	-	0.1

Table 7.1 details the number of senders and of L7 requests on each honeypot. Entries are sorted by the number of senders. Unsurprisingly, the most targeted services are Samba (*smb*), (*http*), Microsoft SQL Server (*mssql*), and Remote Desktop Protocol (*rdp*). For these services, I observe thousands of senders. Interestingly, the number of attempts is clearly disproportional to the number of senders, with RDP attackers generating much more attempts than other cases. Notice the 172 attackers that target the VNC protocol. They sent more than 12 million brute-force login attempts. These figures already shows the heterogeneous picture each honeypot produces.

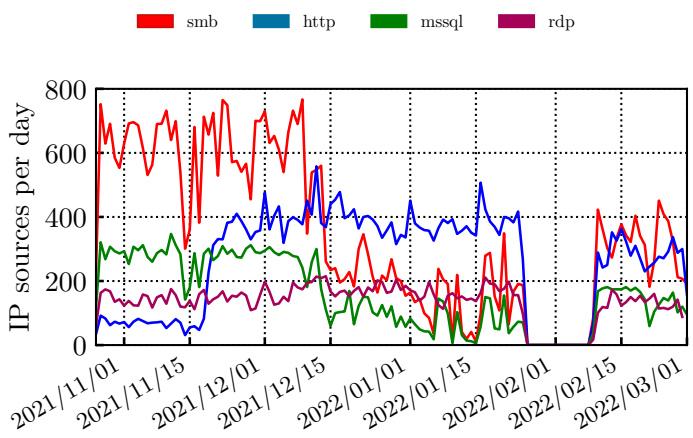


Fig. 7.2 Evolution on number of IP sources for four honeypots.

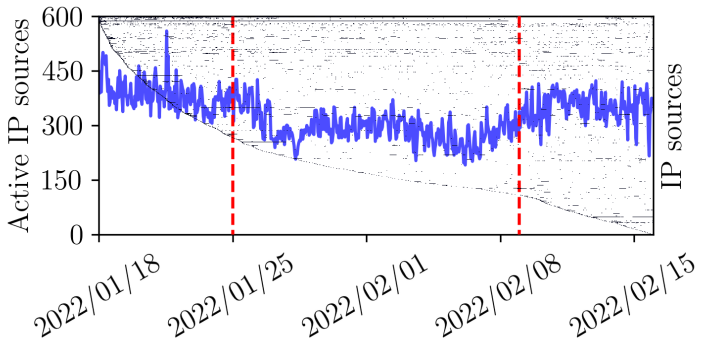


Fig. 7.3 Activity of IP sources probing at application level around the period I shutdown the honeypots.

7.4 Honeypot Traffic Patterns

I report a high-level characterisation of the honeypot traffic to understand overall attacking patterns.

7.4.1 Evolution over Time

I first check the traffic evolution over time for different honeypots. Fig. 7.1 shows the number of source IP addresses seen each day by the services. For readability, services are sorted by popularity. As also observed by others [12, 165] the traffic pattern is highly irregular, with sudden changes and spikes.

Fig. 7.2 details the evolution over time for services with the largest traffic share. Notice for instance the sudden increase of *http* sources in mid-Nov and the sudden decrease of *smb* attackers in mid-Dec. These sudden changes are common, and usually are related to changes in attacking patterns [12].

Focus now on the period when I turn back on the honeypots after the shutdown. I immediately notice a pattern almost similar to the one before the shutdown. Fig. 7.3 details the traffic on the complete set of honeypots. Given all the source IP addresses that generate at least one L7 exchange in the whole time period, I monitor TCP/SYN segments they sent starting from one week before the shutdown, and ending one week after the restart of my infrastructure on the different /24 network.[1]

In Fig 7.3, the red lines highlight the offline period, while the blue line describes the number of IP sources per hour (left y-axis). Each row (right y-axis) refers to a given IP address. Dots are single TCP/SYN packets observed at a given time. IP addresses are sorted by increasing time as they are seen in the infrastructure during this interval. Several considerations hold:

- Attackers return over time, sometimes being active for rather long periods (see dark horizontal segments)
- Old and new attackers continue to search for the honeypots when the systems are offline. The arrival rate is smaller during the shutdown (see the dark external envelope)
- As soon as the systems become active again, attackers immediately discover them – the arrival rate of new IP sources grows again
- No major changes are seen when comparing before and after the shutdown, i.e., attackers get back and keep trying to enter the systems (see also Fig. 7.1)
- The availability of new possible victims attracts new attackers (see the appearance of new attackers that appear starting from Feb. 9th only).

The solid blue line quantifies number of active IP sources per hour. The interest of attackers decreases when the honeypots are unavailable, but it

[1]The traffic *during* the shutdown refers to the initial /24 network.

does not vanish. Recall that here I consider only IP addresses performing L7 interactions at least once, so hosts scanning the network are not counted. While an average of 350 hourly IP sources are seen active before and after the shutdown, I still see around 300 of these attackers active when the infrastructure is offline.

Takeaway Attacking rate is variable over time. Attackers keep returning, even after the honeypots are unreachable for weeks. Attackers' arrival rate grows fast when new honeypots become available.

7.4.2 Popular Sources and Countries

I now check if source IP addresses are associated to well-known actors. For this, I tag source IP address using well-known lists of scanners and crawlers, taken from [174], which include IP addresses of security companies. I also use the well-known Mirai fingerprint to tag sources as Mirai-like node [175].

Tab. 7.2 details the percentage of traffic from these sources to each honeypot. Rows are sorted by popularity. Mirai attackers are the most popular ones (18586 IP addresses in total). They primarily target remote desktop applications that use *rdp* and *vnc*, often performing brute-force password attacks against these services [2]. A significant percentage of traffic to *smb* services comes from Mirai attackers too.

Next, I observe a large quantity of non-malicious sources. For example, I see many requests coming from Google IP addresses on the *http* honeypot, which I associate with Google's crawlers. I also observe traffic from security organisations. Some of them focus on specific services, e.g., Onyphe on *smb* and Stretchoid on *http*. Others scan services horizontally, such as Shadowserver and Shodan. I note that these crawlers do send L7 traffic to test applications, such as trying to login as anonymous in my ftp honeypot.

The remaining sources (42 IP addresses) target mostly the *smb* service, with small percentage of traffic to other services. These are likely bots looking for vulnerabilities.

[2]Mirai is known to scan also for Telnet, ADB and other protocols, which I do not consider in this work.

Table 7.3 Percentage L7 requests per country considering all honeypots.

Traffic Volume					
DE	Germany	19.86%	UK	UK	0.89%
RU	Russia	10.12%	UA	Ukraine	0.70%
US	US	5.62%	IT	Italy	0.67%
LT	Lithuania	4.79%	IR	Iran	0.63%
VN	Vietnam	4.57%	CO	Colombia	0.54%
BR	Brazil	1.41%	PA	Panama	0.15%
PL	Poland	1.05%	HK	Hong Kong	0.12%
CN	China	1.02%	BE	Belgium	0.03%

I also break down the traffic according to the geographic location of source IP addresses. For that, I tag each IP address with its geographic location using MaxMind's GeoIP database [3]. Tab. 7.3 shows the traffic share (number of L7 requests) per country considering all honeypots. The geographic distribution of requests is similar to what is reported in recent previous works [165].

Takeaway Honeypots observe a large volume of non-malicious traffic, coming from crawlers of security companies that do actively test the services (e.g., trying to login). Mirai-like bots still stand among malicious actors.

7.4.3 Vertical vs. Horizontal Activity

Here I quantify if IP sources tend to focus on single service (e.g., in vertical attacks) or multiple services (e.g., in horizontal attacks). Given two services i and j, I extract the set of sources observed for each service, i.e., $S(i)$ and $S(j)$. Then, I compute the *overlap coefficient* defined as the ratio between the intersection of the sets and the size of smallest one:

$$Overlap(i,j) = \frac{|S(i) \cap S(j)|}{\min(|S(i)|, |S(j)|)}$$

[3] https://www.maxmind.com/en/geoip-demo

The *overlap* takes values between 0 and 1, where 0 means the intersection of the two sets is empty, while 1 means that the smallest set is included in the largest one.[4]

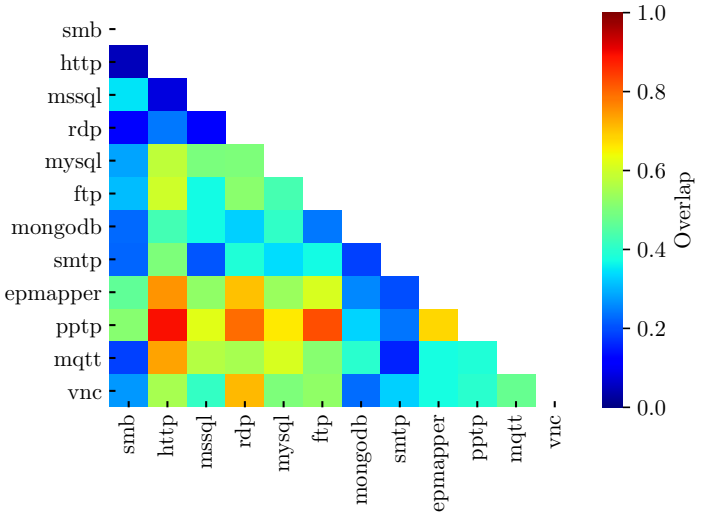


Fig. 7.4 Source overlap among the top-12 services.

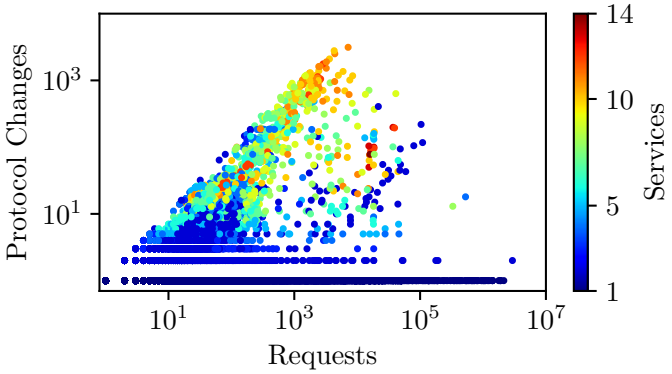


Fig. 7.5 Attackers' activity.

Fig. 7.4 shows a heatmap of the overlap. The warmer the colour, the higher the overlap. Focus on *smb* service (first column). Most of the sources contacting other services are not interested in *smb* (dark blue cells). Exceptions are seen

[4]Given the disparity of sources for each service (see Tab. 7.1) the overlap offers a clearer information than the Jaccard Index.

for those contacting *epmapper, pptp, mssql*: 50% of them do target *smb* too. While this is largely expected for *epmapper* (a service related to Samba), it also suggests attackers search for several Microsoft services at once (e.g., *mssql* and *smb*). Similar pattern is seen for *vnc* and *rdp*, both offering remote login, with overlap at 70%.

The case for *http* is interesting. Most of those contacting the *http* honeypot also contact other services, but not *smb*, *mssql* and *rdp*, all three related to Microsoft services.

I next quantify how many services each source contacts and how often they change the contacted services. For each source, I extract the sequence of contacted services in temporal order considering the entire 5 months of data. I then compute, for each source: (i) the number of times the source changes service; (ii) the total number of unique services it contacts; and (iii) its total number of requests. Fig. 7.5 shows the scatter plot where each dot represents a source. Using log-log scale, the x-axis reports the total number of contacts, the y-axis is the total number of service changes, and colours show the number of unique services per source.

At the bottom, points form a dense horizontal layer. Those are sources that contacted only one service (dark blue), thus never changing protocol (0 changes, artificially reported in the base of the y-log scale). These are *vertical* attackers. Some of these sources contacted my honeypots few times. Others returned millions of times. Manual inspection confirms that these are true attackers focusing on *rdp* and *smb* protocols.

At the second layer of points from the bottom, I see sources that contacted 2 services (light blue). These are attackers that moved from one service to a second and then stayed on this second service (1 change only). A sizeable number of attackers also targeting 2 services (same light blue colour) keep alternating between the 2 services multiple times (more than 2 changes). Here I see the cases of attackers focusing on *categories* of services, such as the remote desktop cases mentioned above.

Finally, I observe multiple sources that keep rotating regularly over multiple services. These are scattered over the diagonal, showing in some case a number of changes proportional to the number of attempts. I here identify horizontal scanners, i.e., sources contacting many services (green to red dots)

and alternating among these services. Some sources are also particularly active, contacting my honeypots thousands of times. I associate this behaviour with some security crawlers, which perform application-layer handshakes to check for service availability – thus, not necessarily performing malicious activities. For instance, those sources who contacted more than 10 services, sending more than 1 000 requests in total (red dots in the top right) are scanners run by security companies, the majority of which managed by *AVAST Software*.

Takeaway Honeypots observe both vertical and horizontal activity. The former is predominately attackers focusing on categories of services. The latter is dominated by scanners and security crawlers.

7.5 Brute-Force Attacks

I now focus on a single type of attack often observed in the honeypots: brute-force password attempts. I first study the used passwords, then I compare attackers' origins and strategies across the various honeypots.

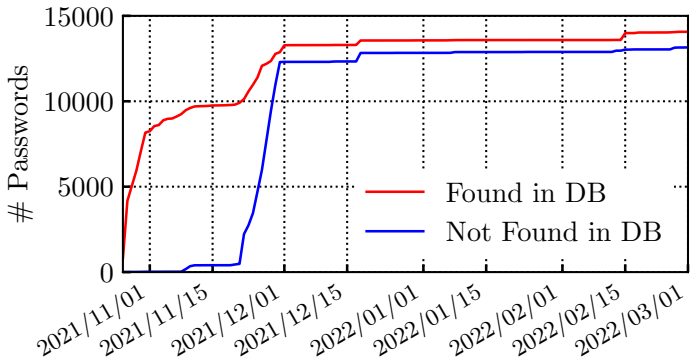
7.5.1 Known vs. Unknown Passwords

Some honeypots in the T-Pot bundle deployed in my network show only the login functionality of services to attackers. The honeypots then save the brute-force attempts, thus allowing us to evaluate the attackers' strategies in terms of used passwords.

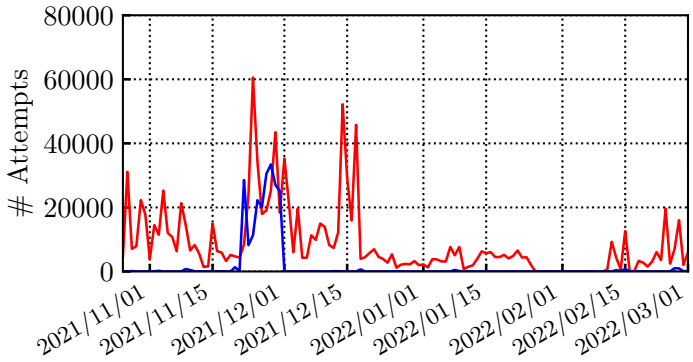
I first verify whether attackers rely on well-known lists of leaked passwords to perform the brute-force attacks. I compare the hashes of passwords seen in my deployment to those in the *Pwned Passwords* project [166].

Fig. 7.6a shows the cumulative number of unique passwords seen in my honeypots. The red line illustrates the number of passwords found in the *Pwned Passwords* database, while the blue line summarises the ones not found in the database.

I observe an increasing number of known passwords (i.e., present in the database) until Nov 10th, 2021. After that date, only few new entries are observed, although attackers still send traffic to the honeypots – see Fig. 7.6b.



(a) Cumulative number of passwords



(b) Attempts per day

Fig. 7.6 Attacks with passwords from known/unknown sources.

In the last week of Nov 2021, more attempts are observed, this time using both new known passwords and passwords that are *not* present in the *Pwned Passwords* database.

Fig. 7.6b extends the analysis by showing the total number of attempts per day using known/unknown passwords. Initially, I record up to 30 thousand attempts per day, all of them using passwords found in the *Pwned Passwords* database. Comparing this figure with Fig. 7.6a, it is clear that attackers keep repeating the same passwords over an over, thus pointing to different actors relying on the same password dictionaries. I see that even during the periods in which few new passwords are observed attackers' activity continue – compare the plateau regions in Fig. 7.6a with the same period in Fig. 7.6b. The figure also reports the start of the attempts with unknown passwords at around Dec

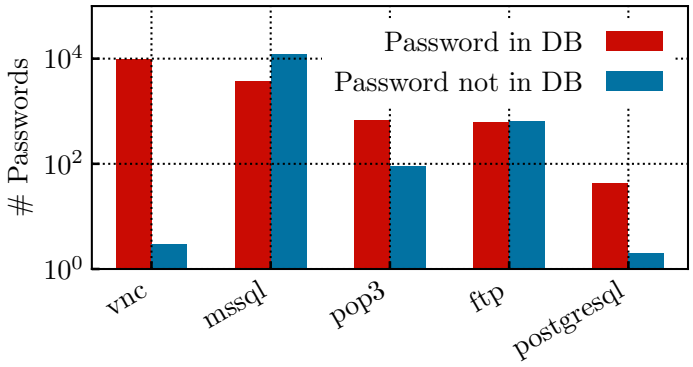


Fig. 7.7 Unknown and known passwords for multiple honeypots.

1st, 2021. Notice how the attempts with such unknown passwords almost vanish after two weeks. I manually inspect this latter set and found that these passwords are probably automatically generated strings.

Finally, in Fig. 7.7 I report the number of unique known/unknown passwords for the five honeypots that save passwords. Both groups of passwords are seen in all honeypots, but in strikingly different proportions. For example, *vnc* has recorded thousands of passwords, almost all of them present in the *Pwned Passwords* database. The *mssql* and *ftp* honeypots, on the other hand, have received more unknown than known passwords overall.

Takeaway Multiple attackers rely on well-known passwords that are seen over and over in various honeypots. Some attackers rely on other lists, not present in well-known password databases. The latter is more common in some honeypots.

7.5.2 Origins of Brute-Force Attacks

Fig. 7.8 breaks down the brute-force password attempts per country. Again, I use the MaxMind's GeoIP database in this analysis. The figure shows different bars for the passwords present in the *Pwned Passwords* database (red) and those not present in the database (blue). Note the y-axis log scale, reporting the percentage of attempts.

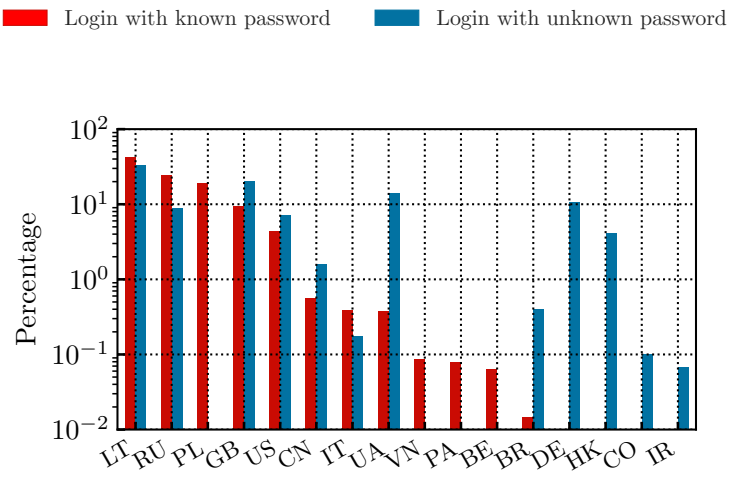


Fig. 7.8 Percentage of password attempts per origin country.

An interesting figure emerges, which is completely different from the overall per-country traffic distribution for all honeypots, reported in Section 7.4.2.

The majority of password attempts in my dataset comes from Lithuania. The country represents 30% of the attempts with a password found in the *Pwned Passwords* database, as well as 20% of the attempts using passwords not in the database. Recalling that Lithuania generates less than 5% of overall traffic observed in the honeypots (cfr. Tab. 7.3), I see that sources in this country are particularly focused on brute-force attacks.

Even more interesting is to observe that *all* brute-force attempts coming from some countries use passwords in the *Pwned Passwords* database (e.g., Poland and Vietnam). Equally, attempts coming from several other countries mainly (or exclusively) use passwords that are not available in the database, i.e., Ukraine, Brazil, Germany, Hong Kong, Colombia and Iran. This result suggests disjoints groups of attackers, with the separation among the groups already visible at country level.

Takeaway The attackers using well-known password databases are clearly distinct from those relying on other lists. This separation is visible even when considering their countries of origin.

7.6 Conclusions

I presented an initial characterisation of the data collected in my horizontal honeypots. I showed how attackers are fast in discovering and trying to abuse the infrastructure. I identified not only groups of attackers performing large-scale attacks against single services, but also those focusing on categories of services or horizontal attempts against all services. I evaluated the passwords used in brute-force login attempts and identified i) attackers relying on well-known password lists; ii) attackers with completely different sets of passwords. The latter ones usually come from different geographic places and focus on particular services.

As future work it would be interesting to extend the infrastructure to other honeypots and locations. I will pursue the creation of *open honeypot datasets*, which would represent an important asset for security analysts and researchers that need to understand cyber-threats and fight attacks. The creation of such open datasets comes with multiple challenges, however. For example, the privacy and security of previous victims must be preserved, as attackers abuse their information to perpetrate new attacks. Finally, would be helpful for the community to use these findings to build updated profiles of active attackers, using automated methodologies to find groups of attacks showing coordinated strategies in the multiple honeypots.

Chapter 8

Conclusions

In this thesis I have presented two fundamental and critical aspects in the context of monitoring modern networks, namely privacy and cybersecurity.

Regarding the privacy problem, my contributions are the following:

- I have engineered an anonymization system (Ch. 2) showing a possible way to simplify and make the data anonymization process more efficient in a live environment, demonstrating that the loss of information is minimal.
- I have generalized the algorithm to apply anonymization with zero-delay (Ch. 3). It is exploitable in many and different fields from network packets to credit card transactions.

These works allowed me to safely and successfully analyze data collected by the teaching servers of the Politecnico di Torino during the Covid-19 health emergency (Ch. 4), presenting a trend of use of the real network that has never been seen before.

Regarding the cybersecurity problem:

- I studied the performance of DPIs (Ch. 5) for building a smart Honeypot.
- I explored the potential of such smart Honeypot: it uncovers attacks that darknets and state of the art active responders would not observe. This "enlightening" brings several benefits and offers opportunities to increase the visibility of sender patterns (Ch. 6).

- I investigated malicious traffic at application level arriving at the Politecnico di Torino network (Ch. 7), observing attackers that always focus on one or few services, and others that target tens of services simultaneously, even practicing brute-force attacks.

As already discussed in the course of this thesis, the first step to be taken to safely perform network monitoring is to anonymize the contents of the packets (Ch. 2 and Ch. 3). To this end, I presented the concept of streaming anonymization which I defined as z-anonymity. This methodology draws inspiration from k-anonymity: often the data collection and the application a posteriori of the algorithm could be not feasible due to the dimensionality of the dataset or to the specific application scenario, so z-anonymity helps in relaxing these constraints. The potential of this lies in its versatility, as it can be applied in any scenario that requires the obfuscation of the streaming data: credit card movements, a new position of the car, or a new visited website. In particular, I focused on the latter scenario, namely the one of network traffic by developing α-MON. It let the user capture network packets through passive probes as if they were already anonymous: it acts transparently simplifying the subsequent data analysis process. I have shown that thanks to its architecture, it is extremely efficient in operations and consequently able to handle the high throughput rates, making network monitoring safe and privacy-preserving.

α-MON has therefore opened the doors to traffic safe analysis in Politecnico di Torino network, in particular I analyzed data obtained from the teaching servers to show the impact of the Covid-19 pandemic on the network (Ch. 4). I showed how the use of the material and the streaming lessons have been a success:

- training has been guaranteed to students from all parts of the world;
- campus network, together with the global one, has been resilient to a notable and sudden increase of data in transit.

In this context, the use of a Big Data approach has made it possible to highlight the evolution of the network projected, at the time of writing, towards a future in which human activities will be strongly linked to its massive use.

This increase in the importance of the network in the economy and in life poses considerable risks and a fertile ground for cybercrime. To give my contribution in the study of cyber attacks and the profiling of attackers, I presented a multidimensional bottom-up approach for network traffic analysis (Ch. 6, Ch. 7): I used Darknets coupled with Honeypots with increasing complexity (L4-Responders, L7-Responders, DPI-Pot), observing their traffic from L3 to L7. With the help, here too, of the technology provided by Big Data approaches, I revisited what the community already knows, bringing some news in terms of methodologies and discoveries. This has allowed to highlight how with the help of such systems it is possible to attract malicious traffic, study it and identify methods to protect systems effectively.

Ultimately with this work I showed that Network Monitoring is possible safely and privacy can be preserved efficiently.

This thesis aims to provide a contribution to the community of researchers around the world who work tirelessly for scientific progress and I hope that the solutions presented here can be a starting point for new methodologies and discoveries. I firmly believe that privacy and cybersecurity issues are of extreme importance and that it is important to invest resources to ensure the safe use of IT tools and promote legality even in the digital world.

References

[1] Internet World Stats.
https://www.internetworldstats.com.

[2] Ericson.com reports and papers.
https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/iot-connections-outlook.

[3] Modern Bank Heists 3.0.
https://www.vmware.com/resources/security/modern-bank-heists-2020.html.

[4] 2021 Norton Cyber Safety Insights Report Global Results.
https://now.symassets.com/content/dam/norton/campaign/NortonReport/2021/2021_NortonLifeLock_Cyber_Safety_Insights_Report_Global_Results.pdf.

[5] European Commission. Regulation (EU) 2016/679 of the European Parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (General Data Protection Regulation), April 2016. Article 1, Subsection: 18, 23, 24, 28, 29, 30, 58.

[6] Cybercrime To Cost The World $10.5 Trillion Annually By 2025.
https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/.

[7] How Mobile ad Fraud has Evolved in the Year of the Pandemic.
https://www.secure-d.io/mobileadfraud2021report/.

[8] A Look Into Gaming and Cybercrime Amid The Pandemic.
https://www.nortonlifelock.com/blogs/gaming/pandemic-gaming-and-cybercrime?_gl=1*1rk9fdv*_ga4_ga*LU41RWVBT2xNS3FPWFZ3bmQzOVE.*_ga4_ga_FG3M2ET3ED*MTY1NTk3Njg1My4yLjEuMTY1NTk3Nzk2Ny42MA..

[9] New Sonicwall research finds aggressive growth in ransomware, rise in IoT attacks.

https://www.sonicwall.com/news/new-sonicwall-research-finds-aggressive-growth-in-ransomware-rise-in-iot-attacks/.

[10] Dawn of the Terrorbit Era. https://www.netscout.com/sites/default/files/2019-02/SECR_001_EN-1901%20-%20NETSCOUT%20Threat%20Intelligence%20Report%202H%202018.pdf.

[11] Alessandro Finamore, Marco Mellia, Michela Meo, Maurizio M Munafo, Politecnico Di Torino, and Dario Rossi. Experiences of internet traffic monitoring with tstat. *IEEE Network*, 25(3):8–14, 2011.

[12] M. Nawrocki, M. Wählisch, T. Schmidt, C. Keil, and J. Schönfelder. A Survey on Honeypot Software and Data Analysis. *arXiv:1608.06249*, 2016.

[13] Apache Hadoop project. https://hadoop.apache.org.

[14] Apache Spark. https://spark.apache.org.

[15] Thomas Favale, Martino Trevisan, Idilio Drago, and Marco Mellia. α-MON: Traffic Anonymizer for Passive Monitoring. *IEEE Transactions on Network and Service Management*, 18(2):1233–1245, 2021.

[16] Thomas Favale, Martino Trevisan, Idilio Drago, and Marco Mellia. α-MON: Anonymized Passive Traffic Monitoring. In *2020 32nd International Teletraffic Congress (ITC 32)*, pages 10–18, 2020.

[17] Alessandro D'Alconzo, Idilio Drago, Andrea Morichetta, Marco Mellia, and Pedro Casas. A survey on big data for network traffic monitoring and analysis. *IEEE Transactions on Network and Service Management*, 16(3):800–813, 2019.

[18] Daniele Apiletti, Elena Baralis, Tania Cerquitelli, Paolo Garza, Danilo Giordano, Marco Mellia, and Luca Venturini. SeLINA: A Self-Learning Insightful Network Analyzer. *IEEE Transactions on Network and Service Management*, 13(3):696–710, 2016.

[19] L. Vassio, D. Giordano, M. Trevisan, M. Mellia, and A. P. Couto da Silva. Users' Fingerprinting Techniques from TCP Traffic. *ACM SIGCOMM Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, 2017.

[20] G. Alotibi, N. Clarke, F. Li, and S. Furnell. User Profiling from Network Traffic via Novel Application-Level Interactions. *International Conference for Internet Technology and Secured Transactions (ICITST)*, 2016.

[21] A. S. Khatouni, M. Trevisan, L. Regano, and A. Viticchié. Privacy issues of ISPs in the modern web. In *2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 588–594, 2017.

[22] J. Xu, J. Fan, M. H. Ammar, and S. B. Moon. Prefix-Preserving IP Address Anonymization: Measurement-based Security Evaluation and a New Cryptography-based Scheme. *IEEE International Conference on Network Protocols (ICNP)*, 2002.

[23] P. Samarati and L. Sweeney. Protecting Privacy when Disclosing Information: k-Anonymity and Its Enforcement through Generalization and Suppression. *Technical Report SRI-CSL-98-04*, 1998.

[24] Stephen Farrell and Hannes Tschofenig. Pervasive Monitoring Is an Attack. RFC 7258, May 2014.

[25] David Naylor, Alessandro Finamore, Ilias Leontiadis, Yan Grunenberger, Marco Mellia, Maurizio Munafò, Konstantina Papagiannaki, and Peter Steenkiste. The Cost of the "s" in https. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 133–140, 2014.

[26] Chia-ling Chan, Romain Fontugne, Kenjiro Cho, and Shigeki Goto. Monitoring TLS adoption using backbone and edge traffic. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 208–213. IEEE, 2018.

[27] Petr Velan, Milan Čermák, Pavel Čeleda, and Martin Drašar. A survey of methods for encrypted traffic classification and analysis. *International Journal of Network Management*, 25(5):355–374, 2015.

[28] P. Hoffman and P. McManus. DNS Queries over HTTPS (DoH). Technical Report 8484, RFC Editor, 2018.

[29] Eric Rescorla, Kazuho Oku, Nick Sullivan, and Christopher A. Wood. Encrypted Server Name Indication for TLS 1.3. Technical Report draft-ietf-tls-esni-04, RFC Editor, 2019.

[30] R. Pang and V. Paxson. A High-Level Programming Environment for Packet Trace Anonymization and Transformation. *ACM SIGCOMM*, pages 339–351, 2003.

[31] M. Peuhkuri. A Method to Compress and Anonymize Packet Traces. *ACM SIGCOMM Internet Measurement Workshop (IMW)*, 2001.

[32] Mark Allman, Ethan Blanton, and Wesley Eddy. A Scalable System for Sharing Internet Measurements. In *Proc. PAM*. Citeseer, 2002.

[33] J. Xu, J. Fan, M. H. Ammar, and S. B. Moon. On the Design and Performance of Prefix-Preserving IP Traffic Trace Anonymization. *ACM SIGCOMM Internet Measurement Workshop*, pages 263–266, 2001.

[34] Hyojoon Kim and Arpit Gupta. ONTAS: Flexible and Scalable Online Network Traffic Anonymization System. In *Proceedings of the 2019 Workshop on Network Meets AI & ML*, page 15–21, New York, NY, USA, 2019. Association for Computing Machinery.

[35] Greg Minshall. Tcpdpriv. http://fly.isti.cnr.it/software/tcpdpriv/.

[36] Ethan Blanton. Tcpurify. https://isc.sans.edu/forums/diary/Truncating+Payloads+and+Anonymizing+PCAP+files/23990/, 2008.

[37] R. Pang and V. Paxson. A High-Level Programming Environment for Packet Trace Anonymization and Transformation. *ACM SIGCOMM Conference*, 2003.

[38] D. Koukis, S. Antonatos, D. Antoniades, E.P. Markatos, and P. Trimintzios. A Generic Anonymization Framework for Network Traffic. In *2006 IEEE International Conference on Communications*, volume 5, pages 2302–2309, 2006.

[39] L. Sweeney. k-anonymity: a Model for Protecting Privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 2002.

[40] Machanavajjhala, Ashwin and Kifer, Daniel and Gehrke, Johannes and Venkitasubramaniam, Muthuramakrishnan. l-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1):3–es, March 2007.

[41] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115. IEEE, 2007.

[42] Adam Meyerson and Ryan Williams. On the Complexity of Optimal K-Anonymity. In *Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '04, page 223–228, New York, NY, USA, 2004. Association for Computing Machinery.

[43] Jianzhong Li, Beng Chin Ooi, and Weiping Wang. Anonymizing Streaming Data for Privacy Protection. In *2008 IEEE 24th International Conference on Data Engineering*, pages 1367–1369, 2008.

[44] Jianneng Cao, Barbara Carminati, Elena Ferrari, and Kian-Lee Tan. CASTLE: Continuously Anonymizing Data Streams. *IEEE Transactions on Dependable and Secure Computing*, 8(3):337–352, 2011.

[45] Intel. Data Plane Development Kit. "https://www.dpdk.org/".

[46] Martino Trevisan, Alessandro Finamore, Marco Mellia, Maurizio Munafo, and Dario Rossi. Traffic Analysis with Off-the-Shelf Hardware: Challenges and Lessons Learned. *IEEE Communications Magazine*, 55(3):163–169, 2017.

[47] Shinae Woo and KyoungSoo Park. Scalable TCP session monitoring with Symmetric Receive-Side Scaling. *KAIST, Daejeon, Korea, Tech. Rep*, 2012. Accessed on 12/13/2016.

[48] Martino Trevisan, Danilo Giordano, Idilio Drago, Maurizio Matteo Munafò, and Marco Mellia. Five Years at the Edge: Watching Internet from the ISP Network. volume 28, pages 561–574. IEEE, 2020.

[49] Benchmarking Methodology for Network Interconnect Devices. RFC 2544, March 1999.

[50] Rolf Neugebauer, Gianni Antichi, José Fernando Zazo, Yury Audzevich, Sergio López-Buedo, and Andrew W. Moore. Understanding PCIe Performance for End Host Networking. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, page 327–341, New York, NY, USA, 2018. Association for Computing Machinery.

[51] Tiancheng Li and Ninghui Li. On the tradeoff between privacy and utility in data publishing. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 517–526, 2009.

[52] Adam Meyerson and Ryan Williams. On the Complexity of Optimal k-anonymity. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 223–228, 2004.

[53] Junwei Zhang, Jing Yang, Jianpei Zhang, and Yongbin Yuan. Kids:k-anonymization data stream base on sliding window. *IEEE International Conference on Future Computer and Communication*, 2010.

[54] Nikjil Jha, Thomas Favale, Luca Vassio, Martino Trevisan, and Marco Mellia. z-anonymity: Zero-Delay Anonymization for Data Streams. In *To appear in the 2020 IEEE International Conference on Big Data*, 2020.

[55] Luca Vassio, Hassan Metwalley, and Danilo Giordano. The exploitation of web navigation data: Ethical issues and alternative scenarios. In *Blurring the Boundaries Through Digital Innovation*, pages 119–129. Springer, 2016.

[56] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-Preserving Data Publishing: A Survey of Recent Developments. *ACM Computing Surveys*, 42(4), June 2010.

[57] Latanya Sweeney. Guaranteeing anonymity when sharing medical data, the Datafly System. In *Proceedings of the AMIA Annual Fall Symposium*, page 51. American Medical Informatics Association, 1997.

[58] Arvind Narayanan and Vitaly Shmatikov. Robust De-anonymization of Large Sparse Datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125, 2008.

[59] Charu C. Aggarwal. On k-Anonymity and the Curse of Dimensionality. In Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi, editors, *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 901–909. ACM, 2005.

[60] Adam Meyerson and Ryan Williams. On the Complexity of Optimal k-anonymity. In *Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '04, page 223–228, New York, NY, USA, 2004. Association for Computing Machinery.

[61] Lada A Adamic, Bernardo A Huberman, AL Barabási, R Albert, H Jeong, and G Bianconi. Power-law distribution of the world wide web. *science*, 287(5461):2115–2115, 2000.

[62] Martino Trevisan, Alessandro Finamore, Marco Mellia, Maurizio Munafo, and Dario Rossi. Traffic Analysis with Off-the-Shelf Hardware: Challenges and Lessons Learned. *IEEE Communications Magazine*, 55(3):163–169, 2017.

[63] Jinliang Fan, Jun Xu, and Mostafa H Ammar. Crypto-pan: Cryptography-based prefix-preserving anonymization. *Computer Networks*, 46(2), 2004.

[64] Nikhil Jha, Luca Vassio, Martino Trevisan, Emilio Leonardi, and Marco Mellia. Practical anonymization for data streams: z-anonymity and relation with k-anonymity. *Performance Evaluation*.

[65] Thomas Favale, Francesca Soro, Martino Trevisan, Idilio Drago, and Marco Mellia. Campus Traffic and e-Learning during COVID-19 Pandemic. *Computer Networks*, 176:107290, 2020.

[66] CloudFlare. On the shoulders of giants: recent changes in internet traffic. https://blog.cloudflare.com/on-the-shoulders-of-giants-recent-changes-in-internet-traffic/, 2020.

[67] Fastly. How covid-19 is affecting internet performance. https://www.fastly.com/blog/how-covid-19-is-affecting-internet-performance, 2020.

[68] SimilarWeb. SimilarWeb coronavirus data & insights hub. https://www.similarweb.com/coronavirus, 2020.

[69] Microsoft. Update #2 on microsoft cloud services continuity. https://azure.microsoft.com/en-us/blog/update-2-on-microsoft-cloud-services-continuity/, 2020.

[70] CNN. Netflix and youtube are slowing down in europe to keep the internet from breaking. https://edition.cnn.com/2020/03/19/tech/netflix-internet-overload-eu/index.html, 2020.

[71] Vodafone. An update on Vodafone's networks, 2020. https://www.vodafone.com/covid19/news/update-on-vodafone-networks.

[72] Telefonica. Operators advise a rational and responsible use of telecommunication networks to cope with traffic increases. https://www.telefonica.com/en/web/press-office/-/operators-advise-a-rational-and-responsible-use-of-telecommunication-networks-to-cope-with-traffic-increases, 2020.

[73] AMS-IX. 17% traffic increase on the ams-ix platform due to corona/covid-19 crisis. https://www.ams-ix.net/ams/news/17-traffic-increase-on-the-ams-ix-platform-due-to-corona-covid-19-crisis, 2020.

[74] NAMEX. Covid-19 emergency: One month after. https://www.namex.it/covid-19-emergency-one-month-after/, 2020.

[75] Comcast. Covid-19 network update. https://corporate.comcast.com/covid-19/network, 2020.

[76] Kenjiro Cho, Cristel Pelsser, Randy Bush, and Youngjoon Won. The Japan Earthquake: the impact on traffic and routing observed by a local ISP. In *Proceedings of the Special Workshop on Internet and Disasters*, pages 1–8, 2011.

[77] Xiaolin Zhuo, Barry Wellman, and Justine Yu. Egypt: the first internet revolt? *Boletim do tempo presente*, (02), 2012.

[78] Jacob Groshek. Forecasting and observing: A cross-methodological consideration of Internet and mobile phone diffusion in the Egyptian revolt. *International Communication Gazette*, 74(8):750–768, 2012.

[79] Alberto Dainotti, Claudio Squarcella, Emile Aben, Kimberly C Claffy, Marco Chiesa, Michele Russo, and Antonio Pescapé. Analysis of country-wide internet outages caused by censorship. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 1–18, 2011.

[80] John Heidemann, Lin Quan, and Yuri Pradkin. *A preliminary analysis of network outages during hurricane Sandy*. University of Southern California, Information Sciences Institute, 2012.

[81] Jan Rüth, Ingmar Poese, Christoph Dietzel, and Oliver Hohlfeld. A First Look at QUIC in the Wild. In *International Conference on Passive and Active Network Measurement*, pages 255–268. Springer, 2018.

[82] Martino Trevisan, Danilo Giordano, Idilio Drago, Maurizio M. Munafò, and Marco Mellia. Five Years at the Edge: Watching Internet From the ISP Network. *IEEE/ACM Transactions on Networking*, 28(2):561–574, 2020.

[83] Michael P Mackrell, Katherine J Twilley, William P Kirk, Luke Q Lu, James L Underhill, and Laura E Barnes. Discovering anomalous patterns in network traffic data during crisis events. In *2013 IEEE Systems and Information Engineering Design Symposium*, pages 52–57. IEEE, 2013.

[84] Renata Teixeira and Jennifer Rexford. Managing routing disruptions in Internet service provider networks. *IEEE Communications Magazine*, 44(3):160–165, 2006.

[85] Yujing Liu, Xiapu Luo, Rocky KC Chang, and Jinshu Su. Characterizing inter-domain rerouting by betweenness centrality after disruptive events. *IEEE Journal on Selected areas in Communications*, 31(6):1147–1157, 2013.

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

[87] Tommaso Rescio, Thomas Favale, Francesca Soro, Marco Mellia, and Idilio Drago. Dpi solutions in practice: Benchmark and comparison. In *Proceedings of the IEEE Security and Privacy Workshops (SPW)*, pages 37–42, 2021.

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

[89] Luca Deri, Maurizio Martinelli, Tomasz Bujlow, and Alfredo Cardigliano. ndpi: Open-source high-speed deep packet inspection. In *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 617–622. IEEE, 2014.

[90] Shane Alcock and Richard Nelson. Libprotoident: Traffic classification using lightweight packet inspection.

[91] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23-24):2435–2463, 1999.

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

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

[94] 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*, pages 81–92. IEEE, 2006.

[95] Marco Mellia, R Lo Cigno, and Fabio Neri. Measuring IP and TCP behavior on edge nodes with Tstat. *Computer Networks*, 47(1):1–21, 2005.

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

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

[98] 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*, pages 956–963, 2013.

[99] Géza Szabó, Dániel Orincsay, Szabolcs Malomsoky, and István Szabó. On the validation of traffic classification algorithms. In *International Conference on Passive and Active Network Measurement*, pages 72–81. Springer, 2008.

[100] Francesco Gringoli, Luca Salgarelli, Maurizio Dusi, Niccolo Cascarano, Fulvio Risso, and KC Claffy. Gt: picking up the truth from the ground for Internet traffic. *ACM SIGCOMM Computer Communication Review*, 39(5):12–18, 2009.

[101] Peng Lizhi, Zhang Hongli, Yang Bo, Chen Yuehui, and Wu Tong. Traffic labeller: collecting internet traffic samples with accurate application information. *China Communications*, 11(1):69–78, 2014.

[102] Hyunchul Kim, KC Claffy, Marina Fomenkov, Dhiman Barman, Michalis Faloutsos, and KiYoung Lee. Internet traffic classification demystified: Myths, caveats, and the best practices. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, New York, NY, USA, 2008. Association for Computing Machinery.

[103] G. Aceto, A. Dainotti, W. de Donato, and A. Pescape. Portload: Taking the best of two worlds in traffic classification. In *2010 INFOCOM IEEE Conference on Computer Communications Workshops*, pages 1–5, 2010.

[104] Jawad Khalife, Amjad Hajjar, and Jesús Díaz-Verdejo. Performance of opendpi in identifying sampled network traffic. *Journal of Networks*, 8(1):71, 2013.

[105] Antonio Nisticò, Dena Markudova, Martino Trevisan, Michela Meo, and Giovanna Carofiglio. A comparative study of rtc applications. *To appear in the Proceedings of the 22nd IEEE International Symposium on Multimedia*, 2020.

[106] 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*, pages 38–45. IEEE, 2006.

[107] Andrea Di Domenico, Gianluca Perna, Martino Trevisan, Luca Vassio, and Danilo Giordano. A network analysis on cloud gaming: Stadia, GeForce Now and PSNow. *arXiv preprint arXiv:2012.06774*, 2020.

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

[109] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Classifying iot devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, 2018.

[110] A Parmisano, S Garcia, and MJ Erquiaga. A labeled dataset with malicious and benign iot network traffic. *Stratosphere Laboratory: Praha, Czech Republic*, 2020.

[111] Francesca Soro, Thomas Favale, Danilo Giordano, Idilio Drago, Tommaso Rescio, Marco Mellia, Zied Ben Houidi, and Dario Rossi. Enlightening the darknets: Augmenting darknet visibility with active probes. *IEEE Transactions on Network and Service Management*.

[112] Claude Fachkha and Mourad Debbabi. Darknet as a source of cyber intelligence: Survey, taxonomy, and characterization. *IEEE Communications Surveys & Tutorials*, 18(2):1197–1227, 2016.

[113] K. Benson, A. Dainotti, K. Claffy, A. Snoeren, and M. Kallitsis. Leveraging Internet Background Radiation for Opportunistic Network Analysis. In *Proceedings of the ACM Internet Measurement Conference*, IMC'15, pages 423–436, 2015.

[114] E. Wustrow, M. Karir, M. Bailey, F. Jahanian, and G. Huston. Internet Background Radiation Revisited. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC'10, pages 62–74, 2010.

[115] A. Dainotti, A. King, K. Claffy, F. Papale, and A. Pescape. Analysis of a "/0" Stealth Scan From a Botnet. *IEEE/ACM Transactions on Networking*, 23(2):341–354, 2015.

[116] Morteza Safaei Pour, Dylan Watson, and Elias Bou-Harb. Sanitizing the iot cyber security posture: An operational cti feed backed up by Internet measurements. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 497–506. IEEE, 2021.

[117] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou. Understanding the Mirai Botnet. In *Proceedings of the 26th USENIX Security Symposium*, USENIX Security'17, pages 1093–1110, 2017.

[118] L. Metongnon and R. Sadre. Beyond Telnet: Prevalence of IoT Protocols in Telescope and Honeypot Measurements. In *Proceedings of the 2018 Workshop on Traffic Measurements for Cybersecurity*, WTMC'18, pages 21–26, 2018.

[119] M. Jonker, A. King, J. Krupp, C. Rossow, A. Sperotto, and A. Dainotti. Millions of Targets Under Attack: A Macroscopic Characterization of the DoS Ecosystem. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, IMC'17, pages 100–113, 2017.

[120] P. Richter and A. Berger. Scanning the Scanners: Sensing the Internet from a Massively Distributed Network Telescope. In *Proceedings of the Internet Measurement Conference*, IMC'19, pages 144–157, 2019.

[121] Francesca Soro, Idilio Drago, Martino Trevisan, Marco Mellia, Joao Ceron, and Jair J. Santanna. Are Darknets All The Same? On Darknet Visibility for Security Monitoring. In *Proceedings of the IEEE International Symposium on Local and Metropolitan Area Networks*, LANMAN, pages 1–6, 2019.

[122] Raphael Hiesgen, Marcin Nawrocki, Alistair King, Alberto Dainotti, Thomas C Schmidt, and Matthias Wählisch. Spoki: Unveiling a new wave of scanners through a reactive network telescope. In *Usenix Security Symposium 2022*. USENIX Association, 2022.

[123] D. Moore, C. Shannon, D. Brown, G. Voelker, and S. Savage. Inferring Internet Denial-of-Service Activity. *ACM Trans. Comput. Syst.*, 24(2):115–139, 2006.

[124] C. Fachkha, E. Bou-Harb, and M. Debbabi. Inferring Distributed Reflection Denial of Service Attacks from Darknet. *Comput. Commun.*, 62(C):59–71, 2015.

[125] A. Dainotti, K. Benson, A. King, B. Huffaker, E. Glatz, X. Dimitropoulos, P. Richter, A. Finamore, and A. Snoeren. Lost in Space: Improving Inference of IPv4 Address Space Utilization. *IEEE Journal on Selected Areas in Communications*, 34(6):1862–1876, 2016.

[126] A. Dainotti, C. Squarcella, E. Aben, K. Claffy, M. Chiesa, M. Russo, and A. Pescape. Analysis of Country-Wide Internet Outages Caused by Censorship. *IEEE/ACM Transactions on Networking*, 22(6):1964–1977, 2014.

[127] Z. Durumeric, M. Bailey, and J. Halderman. An Internet-Wide View of Internet-Wide Scanning. In *Proceedings of the 23rd USENIX Conference on Security Symposiu*, SEC'14, pages 65–78, 2014.

[128] E. Raftopoulos, E. Glatz, X. Dimitropoulos, and A. Dainotti. How Dangerous Is Internet Scanning? A Measurement Study of the Aftermath of an Internet-Wide Scan. In *Proceedings of the 7th Workshop on Traffic Monitoring and Analysis*, TMA'15, pages 158–172, 2015.

[129] S. Staniford, D. Moore, V. Paxson, and N. Weaver. The Top Speed of Flash Worms. In *Proceedings of the ACM Workshop on Rapid Malcode*, WORM'04, 2004.

[130] CAIDA/UCSD. The ucsd network telescope. https://www.caida.org/projects/network_telescope/, 2021.

[131] F. Soro, M. Allegretta, M. Mellia, I. Drago, and L. Bertholdo. Sensing the Noise: Uncovering Communities in Darknet Traffic. In *Proceedings of the Mediterranean Communication and Computer Networking Conference*, MedComNet, pages 1–8, 2020.

[132] GreyNoise. https://greynoise.io/, 2021.

[133] M. Bailey, E. Cooke, F. Jahanian, A. Myrick, and S. Sinha. Practical Darknet Measurement. In *Proceedings of the 40th Annual Conference on Information Sciences and Systems*, CISS'06, pages 1496–1501, 2006.

[134] J. Czyz, K. Lady, S. Miller, M. Bailey, M. Kallitsis, and M. Karir. Understanding IPv6 Internet Background Radiation. In *Proceedings of the 13th ACM Internet Measurement Conference*, IMC'13, pages 105–118, 2013.

[135] Honeynet. The honeynet project. https://www.honeynet.org/, 2021.

[136] TPot. The all in one honeypot platform. https://github.com/telekom-security/tpotce, 2021.

[137] Emiliano De Cristofaro, Arik Friedman, Guillaume Jourjon, Mohamed Ali Kaafar, and M. Zubair Shafiq. Paying for likes? understanding facebook like fraud using honeypots. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14, pages 129–136, New York, NY, USA, 2014.

[138] Steffen Liebergeld, Matthias Lange, and Ravishankar Borgaonkar. Cellpot: A concept for next generation cellular network honeypots. *Internet Society*, pages 1–6, 2014.

[139] Wonkyu Han, Ziming Zhao, Adam Doupé, and Gail-Joon Ahn. 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, page 1–6, New York, NY, USA, 2016.

[140] 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, 2018. USENIX Association.

[141] S. Morishita, T. Hoizumi, W. Ueno, R. Tanabe, C. Gañán, M. J. G. van Eeten, K. Yoshioka, and T. Matsumoto. 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)*, pages 134–143, 2019.

[142] 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*, pages 146–165. Springer, 2004.

[143] Paul Barford, Yan Chen, Anup Goyal, Zhichun Li, Vern Paxson, and Vinod Yegneswaran. Employing honeynets for network situational awareness. In *Cyber situational awareness*, pages 71–102. Springer, 2010.

[144] Eric Alata, Vincent Nicomette, Mohamed Kaâniche, Marc Dacier, and Matthieu Herrb. Lessons learned from the deployment of a high-interaction honeypot. In *2006 Sixth European Dependable Computing Conference*, pages 39–46. IEEE, 2006.

[145] 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)*, pages 1591–1607, Santa Clara, CA, 2019.

[146] 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, pages 0766–0772.

[147] A. Brzeczko, A. S. Uluagac, R. Beyah, and J. Copeland. Active deception model for securing cloud infrastructure. In *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 535–540, 2014.

[148] Cowrie. Ssh/telnet honeypot. https://github.com/cowrie/cowrie, 2021.

[149] Glutton. Generic low interaction honeypot. https://github.com/mushorg/glutton, 2021.

[150] Honeytrap. Advanced honeypot framework. https://github.com/honeytrap/honeytrap, 2021.

[151] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano. nDPI: Open-source High-speed Deep Packet Inspection. In *Proceedings of the International Wireless Communications and Mobile Computing Conference*, IWCMC, pages 617–622, 2014.

[152] Heralding. Credentials catching honeypot. https://github.com/johnnykv/heralding, 2021.

[153] Snare/Tanner. Web application honeypot sensor. http://mushmush.org/, 2021.

[154] Twisted. Event-driven networking engine written in python. https://twistedmatrix.com/trac/, 2021.

[155] Pavol Sokol, Jakub Míšek, and Martin Husák. Honeypots and honeynets: Issues of privacy. 2017(1):4.

[156] Luca Gioacchini, Luca Vassio, Marco Mellia, Idilio Drago, Zied Ben Houidi, and Dario Rossi. Darkvec: Automatic analysis of darknet traffic with word embeddings. In *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '21, page 76–89, New York, NY, USA, 2021. Association for Computing Machinery.

[157] V. Riyadi. Securing mikrotik router. https://mum.mikrotik.com/presentations/ID18/presentation_5554_1540255240.pdf, 2018.

[158] University of Michigan. Why am i receiving connection attempts from the university of michigan? https://cse.engin.umich.edu/about/resources/connection-attempts/, 2013.

[159] ZiHan Wang, ChaoGe Liu, Jing Qiu, ZhiHong Tian, Xiang Cui, and Shen Su. Automatically traceback rdp-based targeted ransomware attacks. *Wireless Communications and Mobile Computing*, 2018, 2018.

[160] Matt Boddy, Ben Jones, and Mark Stockley. Rdp exposed-the threat that's already at your door. *Sophos, Inc, Sophos White Paper*, 2019.

[161] Tim Bai, Haibo Bian, Mohammad A Salahuddin, Abbas Abou Daya, Noura Limam, and Raouf Boutaba. Rdp-based lateral movement detection using machine learning. *Computer Communications*, 165:9–19, 2021.

[162] VirusTotal. https://www.virustotal.com/, 2021.

[163] Thomas Favale, Danilo Giordano, Idilio Drago, and Marco Mellia. What scanners do at l7? exploring horizontal honeypots for security monitoring. In *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 307–313, 2022.

[164] Dionaea. Generic low interaction honeypot. https://github.com/DinoTools/dionaea, 2021.

[165] Raphael Hiesgen, Marcin Nawrocki, Alistair King, Alberto Dainotti, Thomas C Schmidt, and Matthias Wählisch. Spoki: Unveiling a new wave of scanners through a reactive network telescope. *arXiv preprint arXiv:2110.05160*, 2021.

[166] Cloudflare. Pwned passwords. https://haveibeenpwned.com/Passwords, Last visit March 2022.

[167] Zhenxin Zhan, Maochao Xu, and Shouhuai Xu. Predicting cyber attack rates with extreme values. *IEEE Transactions on Information Forensics and Security*, 10(8):1666–1677, 2015.

[168] Gokul Kannan Sadasivam and Chittaranjan Hota. Scalable honeypot architecture for identifying malicious network activities. In *2015 International Conference on Emerging Information Technology and Engineering Solutions*, pages 27–31, 2015.

[169] Harm Griffioen, Kris Oosthoek, Paul van der Knaap, and Christian Doerr. Scan, test, execute: Adversarial tactics in amplification ddos attacks. Association for Computing Machinery, 2021.

[170] Daniel Fraunholz, Daniel Krohmer, Simon Duque Anton, and Hans Dieter Schotten. Investigation of cyber crime conducted by abusing weak or default passwords with a medium interaction honeypot. In *2017 International Conference on Cyber Security And Protection Of Digital Services (Cyber Security)*, pages 1–7, 2017.

[171] Daniel Fraunholz, Marc Zimmermann, Simon Duque Anton, Jorg Schneider, and Hans Dieter Schotten. Distributed and highly-scalable wan network attack sensing and sophisticated analysing framework based on honeypot technology. In *2017 7th International Conference on Cloud Computing, Data Science Engineering - Confluence*, pages 416–421, 2017.

[172] Rdpy. Python implementation of the microsoft rdp protocol. https://github.com/citronneur/rdpy, 2020.

[173] Mailhoney. Smtp low interaction honeypot. https://github.com/phin3has/mailoney, 2021.

[174] Luca Gioacchini, Luca Vassio, Marco Mellia, Idilio Drago, Zied Ben Houidi, and Dario Rossi. Darkvec: Automatic analysis of darknet traffic with word embeddings. In *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '21, page 76–89, New York, NY, USA, 2021. Association for Computing Machinery.

[175] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the mirai botnet. In *26th USENIX security symposium (USENIX Security 17)*, pages 1093–1110, 2017.

Appendix A

Publication, collaborations and projects

In this appendix I report the list of papers published and under review. I also present my collaborations and projects.

A.1 Publications

1. J. Fior, T. Favale, L. Cagliero, D. Giordano, E. Baralis, M. Mellia, D. Moncalvo, P. Baracco, S. Ronchiadin, "*Towards data-driven anti-financial crime: a clustering-based legal entity disambiguator,*". Under review at *IEEE International Conference on Big Data 2022* (The 6th International Workshop on Big Data Analytic for Cyber Crime Investigation and Prevention).

2. S. Geissler, A. Lutu, F. Wamser, T. Favale, V. Vomhoff, M. Krolikowski, D. Perino, M. Mellia, T. Hossfeld, "*Untangling IoT Global Connectivity: The Importance of Mobile Signaling Traffic*,". Under review at *ACM MobiCom* (Annual International Conference On Mobile Computing And Networking).

3. F. Soro, T. Favale, D. Giordano, I. Drago, M. Mellia, T. Rescio, Z. Ben Houidi, D. Rossi, "*Enlightening the Darknets: Augmenting Darknet*

Visibility with Active Probes,". Under review at *IEEE Transactions on Network and Service Management*.

4. T. Favale, D. Giordano, I. Drago and M. Mellia, "*What Scanners do at L7? Exploring Horizontal Honeypots for Security Monitoring*," *2022 IEEE European Symposium on Security and Privacy Workshops* (EuroS&PW), 2022, pp. 307-313, doi: 10.1109/EuroSPW55150.2022.00037.

5. T. Rescio, T. Favale, F. Soro, M. Mellia and I. Drago, "*DPI Solutions in Practice: Benchmark and Comparison*," *2021 IEEE Security and Privacy Workshops* (SPW), 2021, pp. 37-42, doi: 10.1109/SPW53761.2021.00014.

6. T. Favale, M. Trevisan, I. Drago and M. Mellia, "*α-MON: Traffic Anonymizer for Passive Monitoring*," in *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1233-1245, June 2021, doi: 10.1109/TNSM.2021.3057927.

7. T. Favale, F. Soro, D. Giordano, L. Vassio, Z. Ben Houidi, I. Drago, "*The New Abnormal: Network Anomalies in the AI Era*," in *Communication Networks and Service Management in the Era of Artificial Intelligence and Machine Learning*, cap. 11, September 2021, https://doi.org/10.1002/9781119675525.ch11.

8. N. Jha, T. Favale, L. Vassio, M. Trevisan and M. Mellia, "*z-anonymity: Zero-Delay Anonymization for Data Streams*," *2020 IEEE International Conference on Big Data* (Big Data), 2020, pp. 3996-4005, doi: 10.1109/BigData50022.2020.9378422.

9. T. Favale, M. Trevisan, I. Drago and M. Mellia, "*α-MON: Anonymized Passive Traffic Monitoring*," *2020 32nd International Teletraffic Congress* (ITC 32), 2020, pp. 10-18, doi: 10.1109/ITC3249928.2020.00010.

10. T. Favale, F. Soro, M. Trevisan, I. Drago, M. Mellia, "*Campus traffic and e-Learning during COVID-19 pandemic*", *Computer Networks*, Volume 176, 2020, 107290, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2020.107290.

A.2 Research collaborations with research centres, companies and universities

- Associate Researcher in Intesa Sanpaolo (Italy) for development of Antifraud and Anti-Financial Crime systems for hidden risk discovery inside bank transactions.
- Associate Researcher in Huawei Technologies Co. Ltd. (France) for development of solutions to enhance Cybersecurity:
  - Study and development of a new anonymization algorithm.
  - Development of an high-performing anonymization tool for packets capturing.
  - Honeypot infrastructure setup.
  - Analysis of malicious activities on Honeypots.
- Associate Researcher in Telefónica (Spain) for modelling of IoT devices behavior for Cybersecurity purposes through a Big Data and Machine Learning approach.
- Associate Researcher in GARR (Italy) for enhancing cybersecurity and study new network traffic patterns during Covid-19 pandemic.

A.3 Projects

A.3.1 α-MON

Packet measurements at scale are essential for several applications, such as cyber-security, accounting and troubleshooting. They, however, threaten users' privacy by exposing sensitive information. Anonymization has been the answer to this challenge, i.e., replacing sensitive information with obfuscated copies. Anonymization of packet traces, however, comes with some challenges and drawbacks. First, it reduces the value of data. Second, it requires to consider diverse protocols because information may leak from many non-encrypted fields.

Third, it must be performed at high speeds directly at the monitor, to prevent private data from leaking, calling for real-time solutions.

I present α-MON, a flexible tool for privacy-preserving packet monitoring. It replicates input packet streams to different consumers while anonymizing protocol fields according to flexible policies that cover all protocol layers. Beside classic anonymization mechanisms such as IP address obfuscation, α-MON supports z-anonymization, a novel solution to obfuscate rare values that can be uniquely traced back to limited sets of users. Differently from classic anonymization approaches, z-anonymity works on a streaming fashion, with zero delay, operating at high-speed links on a packet-by-packet basis. I quantify the impact of z-anonymity on traffic measurements, finding that it introduces minimal error when it comes to finding heavy-hitter services. I evaluate α-MON performance using packet traces collected from an ISP network and show that it achieves a sustainable rate of 40 Gbit/s on a Commercial Off-the Shelf server. α-MON is available to the community as an open-source project.

A.3.2 DPI-Pot

Darknets collect unsolicited traffic reaching unused address spaces. They bring insights into malicious activities, such as the rise of botnets and DDoS attacks. However, darknets provide a shallow view, as traffic is never answered. I here quantify how their visibility is increased by responding to *some* traffic. To this end, I deploy interactive responders (e.g., honeypots) that can answer unsolicited traffic. Simple at first sight, determining *how* to answer is challenging: From the selection of the protocol to talk to the risk of polluting the collectors with uninteresting data or saturating the infrastructure. I consider four deployments: Darknets, simple L4-Responders, vertical L7-Responders tied to specific ports, and DPIpot, a horizontal honeypot that identifies protocols on the fly on any port. I contrast these alternatives by analyzing traffic attracted by each deployment. I show that interactive responders increase the value of darknet data, uncovering patterns otherwise unseen. I measure *Side-Scan* phenomena in which whenever a host starts responding to a particular port, it attracts traffic to other ports and neighbor addresses. DPIpot unveils attacks that darknets and L7-honeypots would not observe, e.g., large-scale activities on non-standard ports. Some strategies, however, trap senders in certain states, thus hindering

visibility too. Beyond my findings, my analysis can inform the deployment of future monitoring infrastructures combining both darknets and active probes.