Politecnico di Torino

Politecnico di Torino

Cybersecurity

A.a. 2024/2025

Graduation Session October 2025

Cloud-based network telescope: design, deployment and traffic analysis via data mining

Supervisors:

Marco Mellia

Andrea Sordello

Candidate:

Luca Serafini

Summary

Network telescopes, also known as darknets, are passive monitoring sensors widely used by the security community to detect network-related security threats, scanning attempts, and other malicious trends e.g., outbreak of botnet activities. A network telescope consists of an unused range of public IP addresses that do not host any services, so no legitimate traffic should target these addresses. Consequently, the traffic observed to these addresses is by definition caused by unsolicited or anomalous traffic.

Although a significant portion of this unsolicited traffic originates from clearly malicious activities, such as attempts to identify active hosts, open ports, and vulnerabilities, not all unsolicited traffic is inherently malicious. In fact, several legitimate organisations conduct scanning of the entire IPv4 address space for security reasons and for network measurements. Traditionally, organizations that deploy their own custom network telescopes leverage unused ranges of their own public IP address range, and the effectiveness of the telescope is closely related to (i) the size of the organization address range used for host the telescope, (ii) whether any services are hosted on those addresses in the past, and (iii) the geographical location or the AS of the addresses.

In this work, we investigate whether the use of cloud-owned IP addresses can enhance the telescope's coverage respect to the use of traditional organisation-owned IP addresses. This idea is based on the assumption that, since these addresses are widely known to host a wide variety of services, they may be more attractive targets for attackers.

First, we explored several cloud-based solutions for provisioning the resources necessary to deploy the telescope. This step is crucial, as deploying a network telescope can be costly due to the large number of public IPv4 addresses typically required and the structural limitations imposed by cloud providers, such as hidden firewalls that block unsolicited traffic by default or restrictions on the number of IP addresses that can be purchased. In this step, we evaluated the possibility of leasing cloud addresses from a third-party IP broker since it is the cheapest option.

However, this option cannot be considered in our scenario, as the borrowing process changes the autonomous system (AS) and the ownership of the IP, eliminating the distinction of it being a cloud address. Consequently, we evaluated the offerings of three major cloud providers and selected Microsoft Azure, which provided the best balance between cost-effectiveness and ease of deployment, making it the most appropriate choice for our implementation. We conducted tests to determine the limitations imposed by Azure's cloud infrastructure on incoming unsolicited traffic, such as the presence of implicit firewalls or NAT devices that could filter or alter such traffic. Our results revealed the presence of a stateful firewall and the dropping of ICMP timestamp requests. However, this limitation does not affect the telescope's coverage, as the majority of incoming traffic consists of connection attempts. Additionally, there is no filtering based on payload content, as even simple malicious payloads are successfully received by the VM.

Once we set up the Azure environment, we deployed a cloud-based network telescope consisting of 256 IP addresses and monitored it over a period of one month to increase the coverage of our analysis. To gain deeper insights into the activities of Internet scanners targeting the cloud telescope, we do not rely only on passive monitoring but we configured different subnets to periodically host fake services with varying level of interactivity, such as Layer 4 responders and honeypots. These services respond to scanner connection attempts, allowing us to observe and collect the subsequent actions taken by the scanners e.g., the sending of malicious payload, and potentially attract more traffic. The Layer 4 responder is a simple server listening on the entire TCP port range (1-65535), responding to incoming traffic by completing the TCP handshake and then terminating the connection. The honeypot simulated the behaviour of real services by emulating a SSH honeypot i.e., Cowrie on port 22 and an Nginx web server on ports 80 and 443. All remaining subnets functioned as purely passive telescopes, providing no response to incoming traffic. To enable a comparative analysis, we replicated the same setup in a darknet hosted on our campus, leveraging campus-owned IP space.

After we collected the traffic, we performed a statistical analysis focused on identifying differences between the traffic received by the cloud-based network telescope and that targeting a traditional telescope. Specifically, we focused on (i) the traffic volume, (ii), the timeseries of incoming traffic, (iii) the port targeting amplification factor i.e., how much a port is targeted after the activation of interactive responders, and (iv) the diversity of senders. We observed that the cloud telescope captured nearly 2 times the traffic volume and exhibited a higher amplification factor for the Layer 4 responder service, while the honeypot showed more amplification in the traditional telescope. Both the campus and cloud deployments receive traffic from a similar number of unique source IPs, with a

significant overlap in common senders. However, the cloud telescope consistently receives a higher volume of traffic from these shared sources and also attracts a greater number of exclusive senders i.e., sources that contact only the cloud telescope and not the campus deployment.

To delve deeper into the data we collected from both telescopes, we employed data mining techniques to uncover common patterns in the traffic. We used *FP-Growth*, a frequent pattern mining algorithm. We first compared patterns across our different deployment types (darknets, Layer 4 responders, and honeypots). The cloud telescope's darknet revealed significantly more frequent pattern than the traditional telescope, suggesting that attackers conduct more systematic and consistent scans targeting cloud infrastructure. We also observed that interactive services (Layer 4 responders and honeypots) triggered new scanning patterns, revealing how attackers change their behaviour when they found an active host.

However, analyzing the whole telescope traffic produced patterns that were too general due to the enormous diversity of sources and scanning techniques. To address this, we refined our approach by investigating traffic coming from a more restricted groups of hosts e.g., host belonging to the same organizations. We focused on well-known scanning groups, including both malicious actors e.g Mirai and legitimate security platforms e.g. Census. By analyzing each group separately, we uncovered distinct scanning patterns. Some groups exhibited predominant patterns characterized by consistent combinations of targeted ports, while others displayed dynamic behaviors, such as sequential scanning. Additionally, some scanners demonstrated memory-effect patterns, revisiting previously targeted ports to check for changes in their status.

In conclusion, this work demonstrates that deploying a network telescope based on cloud-owned IP addresses offers several advantages over traditional approaches e.g, major volume of traffic, demostrating that these addresses are more exposed than traditional organisation-owned IP.

Although applying data mining techniques to the entire traffic dataset revealed only a few generic patterns, focusing on specific subsets of senders enables the extraction of more precise insights, uncovering the distinct scanning behaviors adopted by different sender groups.

All in all, these findings highlight the potential of cloud-based network telescopes as a valuable resource for the research community in studying cybersecurity events within Internet traffic.

Acknowledgements

Desidero esprimere in questo spazio la mia gratitudine a tutte le persone che hanno reso possibile il completamento di questa tesi.

Il più grande ringraziamento va alla mia famiglia per avermi permesso di intraprendere questo percorso di studi, e portare a termine questa esperienza di crescita umana e professionale. Vorrei dedicare questo momento importante della mia vita ai miei nonni, che non hanno potuto vedermi oggi, ma che porto sempre nel cuore e nei miei pensieri.

Un ringraziamento va al mio relatore, Prof. Marco Mellia, e al Prof. Idilio Drago per la loro guida e supporto durante tutte le fasi di questo lavoro. Ringrazio il mio correlatore, Andrea Sordello, al quale sono grato per i preziosi suggerimenti, il confronto costante e la grande disponibilità dimostrata. Un grazie speciale va anche al gruppo di ricerca SmartData@PoliTO per avermi accolto in un ambiente stimolante e collaborativo.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Context

The Internet is a vast network of interconnected devices worldwide, and the traffic observed on it is not solely the result of legitimate connections between users and services. A significant portion consists of unsolicited traffic generated without any prior request or user interaction, commonly known as Internet Background Radiation (IBR) [1]. IBR is pervasive and reaches any device with a public IP address. One of its primary sources is network scans, which serves to identify active hosts, open services, and potential vulnerabilities, making them valuable tools for attackers. However, not all unsolicited traffic is malicious. Internet scanning platforms also use network scanning to conduct security researches and map devices present on the Internet.

Monitoring IBR is essential for understanding the techniques and patterns employed by attackers, as well as for identifying trends and behaviors among groups of hosts performing scans. Network telescopes are widely used systems designed to monitor IBR. They observe traffic directed toward unused portions of the IP address space, where no services are hosted, thus ensuring that all incoming traffic results from unsolicited activity. Not all network telescope are the same. Their effectiveness depends on several factors, including the size of the monitored address space, its previous usage, and the associated Autonomous System (AS).

Network telescope are typically hosted within traditional network infrastructures, such as university campus or research institutions. However, several works explore their implementation within cloud environments. Cloud computing presents a new frontier for network telescopes, as cloud providers manage vast IP address spaces. Traffic directed toward these addresses may exhibit unique characteristics compared to traditional network telescopes, offering new opportunities for analysis and research.

The large volume of data collected by network telescopes presents challenges in extracting meaningful insights from IBR to identify patterns and scanner behaviors. In this context, data mining techniques offer valuable tools. By identifying frequent patterns, such as commonly targeted destination ports, these techniques enable comprehensive characterization of both overall internet scanning activity and the behavior of specific host groups.

1.2 Research questions

In this section we list the key questions to be addressed throughout the thesis:

- **Can we deploy a network telescope in a cloud environment?**

  We investigated the possibility of mounting a network telescope within a cloud environment to monitor IBR targeting cloud-owned IP addresses. This involved addressing several sub-questions:

  - **What are the possible solutions?**

    We investigated the main strategies for obtaining a subnet address space and associating it with a Virtual Machine (VM) to deploy the network telescope. We evaluated the costs proposed by major cloud providers (AWS, GCP, Azure) and analyzed the constraints imposed by each. Additionally, we explored the option of IP leasing by renting a subnet from an IP broker. We present our findings in Chapter 3.

  - **Does the cloud environment bring limitation for hosting a network telescope?**

    Given that the traffic is collected from a non-traditional environment, we investigated whether the cloud provider's infrastructure imposes any undeclared restrictions on unsolicited traffic that could potentially limit the observation of IBR. We performed several tests to evaluate the feasibility of embedding a network telescope within a cloud environment, focusing on identifying any restrictions imposed by the cloud provider's infrastructure investigating the presence of any traffic filtering or transformation. Chapter 4 presents the test results.

- **Do we observe differences of traffic targeting a cloud telescope compared to traffic targeting a traditional one?**

  We compared the traffic collected by the cloud telescope with that of a traditional network telescope hosted on our campus IP space. We performed a statistical analysis to address the following sub-questions:

  - **How different is the traffic of a cloud telescope compared to a traditional one? Does the cloud telescope attract more senders?**

    We compared the traffic collected by both telescopes over a one-month period, focusing on traffic volume and the time series of incoming flows. Additionally, we analyzed sender diversity by comparing the unique source IPs contacting each telescope, examining flow distributions and identifying common and exclusive senders. The results of this comparison are detailed in Chapter 5.

  - **How does the activity of IBR sources change when we interact with them? Does the same behavior occur in a traditional telescope?**

    We evaluated the impact of activating several interactive services, e.g., Layer 4 responder and honeypot, on predefined subnets within both the cloud and traditional telescopes. We analyzed how this change affected the volume and characteristics of incoming traffic, comparing the amplification effects observed on both telescopes under different interactivity configurations. We present the results of this comparison in Chapter 5.

- **Can data mining extract traffic patterns from traces of a cloud-based network telescope?**

  We applied data mining techniques to extract patterns from the traffic collected by both telescopes to address the following sub-questions:

  - **Is there any global common pattern? Do traffic patterns change between cloud and traditional telescope?**

    We considered the traffic generated by the overall internet scanning activity across different deployment types (e.g., darknets, Layer 4 responders, and honeypots), extracting frequent patterns to identify general trends and behaviors. We compared the patterns observed in the cloud-based telescope with those from the traditional campus-based telescope. The results of this analysis are presented in Chapter 6.

  - **How well-known sources of IBR conduct their activities?**

    We analyzed subsets of traffic generated by well-known groups of hosts, including both malicious actors (e.g., Mirai) and legitimate scanning platforms (e.g., Census). This analysis focused on uncovering their specific scanning behaviors and identifying characteristic patterns. The results of this analysis are presented in Chapter 6.

1.3 Related Work

This section highlights the key previous works related to this thesis. The literature includes numerous studies on the adoption of network telescopes, with some focusing specifically on cloud-based telescopes. Additionally, several works explore the analysis of scanning activities using data mining techniques.

1.3.1 Network Telescope in the Cloud

Several studies have investigated the deployment of network telescopes to monitor IP addresses belonging to cloud providers, enabling comparisons with traditional telescopes.

In [2], the authors proposed a geographically distributed, cloud-native network telescope architecture that captured five months of traffic across twenty-six regions worldwide. The design leveraged general-purpose virtual machines offered by AWS under the *spot* pricing model, providing a cost-effective alternative to the *on-demand* pricing model. However, *spot* instances are subject to interruptions by the Cloud Service Provider (CSP) with a two-minute warning, either due to higher bids from other clients or temporary shortages of *spot* instances. To address this, the authors implemented an efficient replacement mechanism to ensure data preservation and promptly replace interrupted instances with new ones. However, this results in a new public IP address reassignment from the CSP address pool, thus contributing to a wider diversity of observed addresses. The cloud telescope utilized an IP address range varying from as small as "one IP address per each of the 26 available cloud regions" (26 IP addresses) to as large as "ten IP addresses per available cloud region" (260 IP addresses). The results demonstrated a higher volume of network packets per IP compared to traditional telescopes, with TCP traffic being the most prevalent (80.18%), followed by ICMP (16.22%) and UDP (3.58%). Overall, the authors highlighted the advantages of cloud-based telescopes, including their cost-effectiveness, scalability, and extensive coverage. The global distribution of their approach allowed for the collection of traffic targeting a diverse range of countries and regions, offering the potential to uncover intriguing geopolitical patterns.

Another notable study is DSCOPE [3], which also utilized the *spot* instance solution offered by AWS, similar to the previous work. Due to the random availability of addresses and instances across regions, the authors normalized the collected data by subsampling 750 unique IP addresses per region per day, with a total of 125 IP-hours of data collected per day. The analysis focused on comparing the observed traffic with that of a traditional telescope. Notably, the results revealed that cloud IP traffic is more variable than that of traditional network telescopes and receives traffic from a significantly larger number of IP addresses. The number of scanners varies by region, and the scans are predominantly untargeted.

1.3.2 Characterizing internet activity with data mining

Data mining techniques present a useful tool for analyzing and getting insights from large scale datasets. Several works leverage their application to characterize the behaviour of malicious hosts performing scans.

In [4], the authors applied an association rule mining algorithm to uncover patterns in the behavior of well-known malware families targeting Internet of Things (IoT) devices. Their analysis focused on generating rules based on specific indicators, such as *destination port*, *TCP window size*, and *type of service*, in traffic generated by malware like *Mirai* and *Hajime*. They applied the algorithm to daily datasets, grouped by source IP activity, and produced three sets of association rules. Each set characterized the packets of a source IP according to the contacted ports, the TCP window size, or the type of service used. To reliably identify hosts infected by *Mirai* and *Hajime*, the authors relied on known fingerprints, which allowed them to estimate with 90% probability whether a packet originated from an infected host. This improved the accuracy of identifying infected machines within the broader Internet Background Radiation. Figure 1.1 illustrates a timeline of association rules derived for each of the three indicators for hosts affected by the *Mirai* malware. Notably, the authors observed how the number of hosts matching different rules evolved over time, particularly during the *Mirai* outbreak.

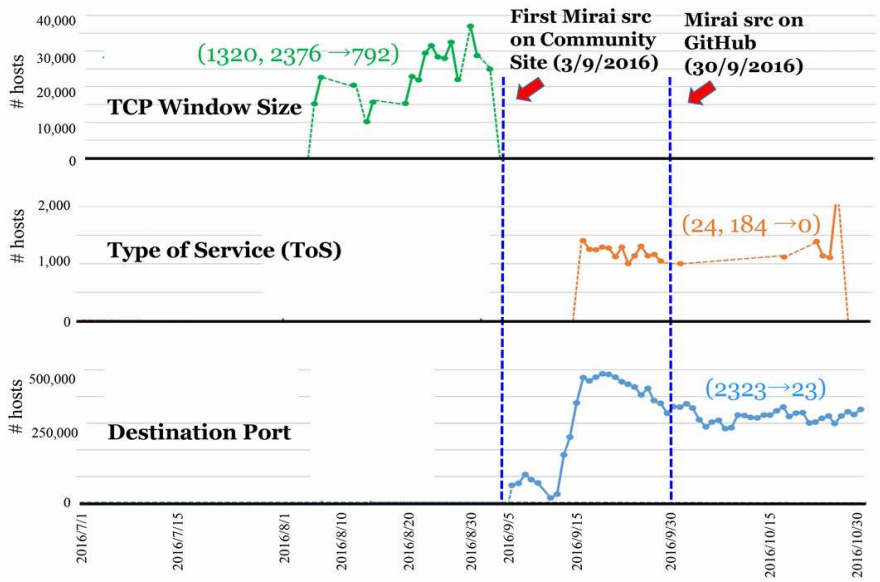


Figure 1.1: Timeline of extracted association rules for destination ports, TCP window size, and type of service from Mirai-infected hosts [4].

1.4 Contributions

In this section, we list the main contributions of this thesis.

- **Exploration of cloud providers for hosting a network telescope**. We explored various solutions for acquiring the resources needed to deploy a network telescope in a cloud environment. Our investigation included evaluating the acquisition of IP address space from major cloud providers and the option of leasing a subnet from an IP broker. For each solution, we assessed the associated costs and constraints. Then, we evaluated whether the provisioned cloud infrastructure allows complete observation of IBR without interference from undeclared firewalls or NAT devices that could filter or alter unsolicited traffic. To validate this, we put the cloud VM under several scanning tests, monitoring for any packet loss or alteration in the network traces.

- **Cloud IP deployment and interactive services implementation**. To accommodate the large IP address space required for our network telescope, we developed a script to automate the deployment of IP addresses. Additionally, as part of the interactive services deployed within specific subnets, we present the implementation of a Layer 4 responder.

- **Comparison of cloud and traditional telescopes**. We conducted a statistical analysis of the traffic targeting both a cloud-based and a traditional network telescope. We made a comparison focused on traffic volumes, time-series of incoming flows, diversity of senders, and the amplification effects observed due to the activation of fake interactive services.

- **Impact of interactivity levels**. We analyzed how traffic targeting a network telescope changes when specific subnets are configured to host fake services with varying interactivity levels, such as Layer 4 responders and honeypots. We compared the amplification effects observed on both cloud-based and traditional telescopes under different interactivity configurations, evaluating these effects using the port amplification factor.

- **Data mining techniques to analyze scanning activities**. We applied frequent pattern mining to analyze scanner activity targeting both cloud and traditional network telescopes. Our focus was on identifying general trends and patterns of internet scanning across specific subnets with varying levels of interactivity (e.g., Layer 4 responder and honeypot). Additionally, we examined well-known host groups, both malicious (e.g., Mirai) and benign (e.g., legitimate scanning platforms), to uncover their distinct scanning behaviors and characteristic patterns.

1.5 Index

In Chapter 2, we introduce the concept of Internet Background Radiation (IBR), explaining the role of network telescopes and the fundamentals of cloud computing. We also provide an overview of data mining techniques useful for analyzing large-scale datasets.

Chapter 3 explores various strategies and solutions for implementing a cloud-based network telescope, focusing on approaches involving major cloud providers (AWS, GCP, Azure) and the IP leasing market.

In Chapter 4, we detail the implementation of a cloud-based network telescope within the Azure cloud environment. We begin by conducting tests to evaluate potential restrictions imposed by the cloud infrastructure on observable traffic. Next, we describe the deployment process of the cloud telescope, including the allocation of interactive responders on specific subnets.

Chapter 5 presents the data collection campaign conducted on both traditional and cloud-based network telescopes, followed by a statistical analysis of the collected data to highlight differences in traffic targeting the two telescopes.

In Chapter 6, we apply frequent pattern mining techniques to identify overall Internet scanning patterns and to characterize the behavior of specific groups of hosts performing scans.

Finally, in Chapter 7, we summarize the main findings of this thesis.

Chapter 2

Background

2.1 Internet Background Radiation

Network traffic is constant and pervasive, but not all of it is legitimate or productive. In 2004, Pang et al. [1] introduced the term "Background Radiation" to describe unsolicited and nonproductive traffic. This type of traffic is often directed at nonexistent addresses, inactive servers, or servers that do not intend to receive it. Broadly, the intent behind such traffic can be either malicious or benign. Literature propose different types of classification for characterizing traffic constituting IBR [5]. In general, three main categories exist:

- **Network scanning**. It is the process of identifying active hosts, open ports, and other critical information within a network. It helps assess network security, detect vulnerabilities, and maintain network health and performance. Attackers often utilize automated tools to conduct scans with the aim of uncovering vulnerable hosts or exposed services on the Internet, such as open ports or outdated software versions.

- **Backscatter**. This refers to unsolicited traffic resulting from denial-of-service (DoS) attacks, where attackers spoof the source IP address to obscure the origin of the attack. By spoofing the source IP address, they cause the victim to send responses to these forged addresses, leading to a flood of unsolicited traffic [6].

- **Misconfigurations**. These originate from network errors or misconfigured systems that unintentionally direct traffic to incorrect destination addresses, causing hosts assigned to these addresses to receive unsolicited traffic.

2.2 Network scanning

Network scanning is the process of discovering active hosts on the network and gathering information about them, such as operating system, active ports, services, and applications. The following techniques form the core of network scanning [7]:

- **Host Discovery**. This is the initial step in network scanning. It involves sending messages to a host to elicit a response, which indicates the host is active. However, host discovery methods are not always reliable. A lack of response may suggest that the target is inactive, but it could also mean that a router or firewall is blocking or dropping the packets.

- **Port and Service Scanning**. This involves sending messages to a host to determine which ports are open and which services are running on those ports. Knowing the open ports and services helps attackers further investigate vulnerabilities that can be possible entry points into the system. There are many different types of port scanning techniques. Some of them are:

  - **TCP Connect Scan**. A TCP Connect Scan is the simplest type of port scan. It attempts to establish a full TCP three-way handshake with the target port. If the port is open and not blocked by a firewall, the target responds with a SYN/ACK packet. If the port is closed, the target responds with a RST/ACK packet.
  - **Half-Open Scan**. A half-open scan, also known as a SYN scan, is a more stealthy scanning technique that does not complete the full TCP three-way handshake. In this method, the scanner sends a SYN packet to the target port. If the target responds with a SYN/ACK packet, it indicates that the port is open. Instead of completing the handshake, the scanner immediately terminates the connection by sending a RST packet.
  - **Stealth scan**. Stealth scans use a variety of evasive techniques to evade detection by firewalls and intrusion detection systems. These scans send packets with unusual TCP flag combinations or fragmented packets to probe ports without establishing a full connection or triggering alarms. Common stealth scan types include: SYN/ ACK scan, FIN scan, ACK scan, NULL scan, and XMAS (Christmas Tree) scan.

  As with host discovery, routers and firewalls may represent a problem for port scanning as they may block or drop packets.

- **OS Detection**. OS detection, also known as OS fingerprinting, involves sending specially crafted messages to an active host to elicit specific responses that reveal the type of operating system running on the host.

Many network scanning tools exist, the most common is Nmap [8], an open source tool for network exploration and security auditing.

2.2.1 Who is scanning?

Internet scanning has become a prevalent activity on the Internet. Griffioen et al. [9] conducted a study on internet-wide scanning over a 10-year period (2015-2024), revealing a 30-fold increase in scanning activity during this time. In general, network scanning activities involve a wide range of actors and can result from both malicious or legitimate purposes. While attackers use scanning to identify and exploit vulnerable hosts, it also serves as a valuable tool for internet scanning platforms to conduct security research and network measurements.

Mirai Botnet

Malicious scanning is often the result of coordinated activities by attackers or automated systems. Botnets, in particular represent one of the most common sources of malicious scanning activity. A botnet is a network of compromised computers or devices controlled by a central entity, often referred to as a "botmaster." The botmaster leverages these compromised devices, known as "bots" or "zombies," to execute various malicious activities, including distributed denial-of-service (DDoS) attacks, spamming, phishing, and network scanning. Botnets are among the most prevalent tools for cybercriminal activities. They perform network scans for multiple purposes, such as identifying vulnerable machines to infect and recruit into the botnet or probing networks for enumeration and penetration. A common scanning technique employed by botnets is horizontal scanning, where they systematically probe the same protocol port across large ranges of IP addresses, often targeting random IP addresses [10].

A notable example of botnet is the Mirai botnet [11]. Mirai is a notorious malware family that specifically targets Internet of Things (IoT) devices, incorporating them into a powerful DDoS botnet. Mirai propagates by scanning random IPv4 addresses on Telnet TCP ports 23 and 2323, attempting brute-force logins using a predefined list of weak or default credentials. Once a device is successfully compromised, the Mirai bot reports the victim's IP address and credentials to a centralized report server. A separate loader program then infects the device by downloading an architecture-specific malware binary, completing the compromise process.

"MalwareMustDie," a white-hat security research group, first identified Mirai's activities in August 2016 [12]. The malware's source code [13] was publicly released in September 2016, which triggered the rapid emergence of numerous Mirai variants [14]. Analysis of the released source code revealed that during the scanning phase of Mirai's workflow, bots send TCP SYN packets to random IP addresses.

Notably, it also revealed a fingerprint: the TCP sequence number is set equal to the destination IP address (i.e., TCP.seq == IP.dst), with the probability of this occurring randomly being $1/2^{32}$.

Internet Scanning Platforms

Network scanning is not always driven by malicious intent. Various organizations and platforms perform internet-wide scans for purposes such as security research and network measurements. Several scanning platforms exist and operate with specific objectives. Examples include:

- **Censys**. It is a public search engine i.e., a system that continuously scans the Internet to discover and identify active devices connected to IP networks. Its foremost goal is to maintain a comprehensive and up-to date map of Internet devices, services, web properties (i.e., websites), and certificates. Censys collects data on hosts and websites through periodical and horizontal scan of the IPv4 address space [15].

- **Census**. The Internet Census Group [16] analyzes trends and benchmarks security performance across various industries. Their regular Internet scans aim to identify outdated software and security vulnerabilities that attackers could exploit.

- **Binaryedge**. This is a distributed platform comprising of scanners and honeypots sensors that continuously collect, classify and correlate different types of data from the internet. They create reports of what is present on the internet, and estimate what is an organization's attack surface [17].

- **Hinet**. Hinet.net [18] is a major Taiwanese internet service provider (ISP) operated by Chunghwa Telecom, providing a variety of online services including broadband internet access, web hosting, cloud solutions, and security offerings. Though primarily known as an ISP, Hinet also offers security-related services such as vulnerability assessments and DDoS protection that involve network scanning to identify security weaknesses and protect infrastructure.

- **Stretchoid**. Stretchoid's activity is occasionally flagged as malicious by security systems such as firewalls. However, the platform describes itself as a benign service aimed at assisting organizations in identifying their online services through internet-wide scanning [19].

- **Shadowserver**. The Shadowserver Foundation [20], a non-profit organization, performs daily scans of the entire IPv4 address space. Their mission is to improve internet security by identifying vulnerabilities and threats and providing detailed reports to their subscribers.

2.3 Monitoring Internet Background Radiation

Monitoring Internet background radiation enables researchers to characterize the nature and volume of unsolicited traffic, as well as to analyze large-scale attacks and other global events [21]. However, a basic issue when attempting to measure IBR is how, across all observed traffic, to determine which is indeed unwanted. Simply including all unsuccessful connection attempts, one can mix truly unwanted traffic with traffic representing benign, transient failures, such as accesses to Web servers that are usually running but happen to be off-line during the measurement period. Instead, by measuring only incoming traffic targeting non-existent services, IPv4 addresses that are unallocated or unused, we can filter out most forms of benign failures and focus on traffic that is highly likely to represent unwanted activity [1]. The security, operations, and research communities widely monitor allocated, globally routable, but unused IPv4 address blocks to study various Internet phenomena [22]. Since no active hosts exist in these unused blocks, any packets sent to these IP addresses result from unsolicited activity.

Systems that monitor unused address spaces have a variety of names, including darknets, network telescopes, blackhole monitors [23], network sinks [24], and network motion sensors [25]. From now on, we will refer to these systems as Network Telescopes.

2.3.1 Network Telescope

The term "Network Telescope" refers to a passive sensor system designed to collect incoming IBR within a segment of unused IP address space [26]. The name draws an analogy to astronomical telescopes, which observe the universe by capturing photons through their apertures. Similarly, a network telescope captures packets arriving at a portion of monitored address space. While astronomical telescopes are useful for studying celestial objects and their properties, network telescopes are adopted to analyze host behavior and characteristics, such as activity patterns, intensity, or categorization (e.g., denial-of-service attacks or worm infections).

In essence, a network telescope consists of a device that combines hardware and software components to receive traffic routed to it. The figure 2.1 illustrates a basic network telescope topology. In this setup, the router directs productive traffic through the firewall towards the Local Area Network where active addresses exist, while it forwards the unused portion of the IP address range to the network telescope [2]. To collect unsolicited traffic, the network telescope must perform a capture of incoming packets. This requires a process with sufficient privileges to bind to the raw packet capture interface provided by the operating system. `tcpdump` [27] is a commonly used tool for traffic capture, which leverages the `libpcap` library to obtain copies of raw incoming traffic directed to the network interface connected

to the Internet. The result is a file in the PCAP format containing network packets with all their relevant fields (`src_ip`, `dst_ip`, `src_port`, `dst_port`, `proto`, etc.).

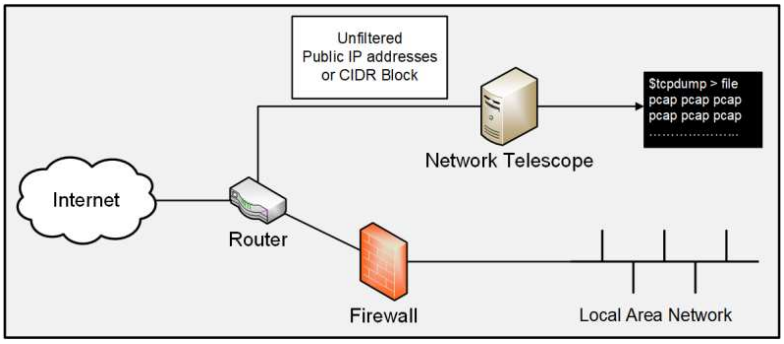


Figure 2.1: Basic Network Telescope Topology [2]

The effectiveness of a network telescope largely depends on the size of the monitored address space. The larger the address space, the higher the probability of capturing specific events. We can express this probability as the ratio between the size of the monitored address space and the total size of the IPv4 address space. For instance, monitoring a /24 subnet (256 addresses) provides a probability of $\frac{1}{2^{24}}$ of capturing a specific event, while monitoring a /8 subnet (16,777,216 addresses) increases this probability to $\frac{1}{2^8}$.

Many network telescope implementations exist. One of the most notable is the CAIDA Network Telescope [28]. This telescope monitors traffic directed to a /9 and /10 IPv4 network, which represents approximately 1/256th of the entire IPv4 address space. The data collected by the CAIDA Network Telescope serves as a valuable resource for the research community in characterizing IBR, contributing to more than 300 publications.

Honeypot

Network telescopes are generally passive sensor systems that only receive traffic without providing any response. However, other variants exist that involve some level of interaction with incoming traffic. These sensors are often referred to as honeypots. These can be classified based on their level of interactions [29]:

- **Low interaction**. These implement the general behaviour of a Layer 4 responder. A low-interaction system enables observation of data payloads in the first TCP packet after completing the handshake. The system responds to incoming TCP requests, completes the handshake, and drops the connection after capturing the first data packet.

- **Medium interaction**. These implement the behaviour of a general honeypot system, implementing portions of selected application layer protocols, such as HTTP's GET method, or FTP's list command, to obtain TCP payloads beyond the first packet, to learn what exploiting actions attackers would take upon successful connection to the target.

- **High interaction**. In this variant, the sensor expose a virtualised implementation of intentionally vulnerable services, with the objective to observe the infection commands taking place beyond transport-layer handshake, i.e. during protocol message exchange and data/payload transfer.

Distributed Network Telescope

Moore et al. [26] discussed some practical issues in the use of a network telescope for detecting and measuring events on the Internet. One main limitation is that in the reality attackers do not perform scanning in a uniform manner across the entire address space. Instead, they may exclude specific address regions, scan certain areas more heavily, or rely on flawed pseudo-random number generators (PRNGs) that introduce bias into the scanning process. This non-uniform scanning behaviour can lead to misrepresent the actual activity in the Internet, as some address ranges may be over-represented while others are under-represented or not represented at all.

To address these limitations, Moore et al. proposed the use of a distributed network telescope. This includes the combination of multiple network telescopes, each monitoring a different portion of network space of different subnetworks, to create a larger telescope that observe a wider range of IP addresses. A distributed telescope offers several advantages. By covering a larger portion of the address space, it enhances detection time, duration, precision, and provides more reliable estimates of targeting rates compared to individual telescopes. Monitoring diverse and geographically spread address ranges help reveal and overcome targeting bias in the event. Additionally, combining the resources of multiple telescopes increases overall network capacity and monitoring capabilities, making it easier to manage high traffic volumes during peak events.

2.4 Cloud Computing

The National Institute of Standard Technology (NIST) [30] defines cloud computing as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal

management effort or service provider interaction. Some of the key characteristics of cloud computing include:

- **On-demand self-service**. Users can automatically provision computing capabilities, such as server time and network storage, without requiring human interaction with each service provider.
- **Broad network access**. Capabilities are available over the network and can be accessed through standard mechanisms that promote use across various platforms (e.g., mobile phones, tablets, laptops, and workstations).
- **Resource pooling**. The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. This includes storage, processing, memory, and network bandwidth.
- **Rapid elasticity**. Capabilities can be elastically provisioned and released to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear unlimited and can be appropriated in any quantity at any time.
- **Measured service**. Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

Many companies rely on cloud computing services due to their flexibility, scalability, and cost-effectiveness. Major public cloud service providers, such as Amazon Web Services (AWS) [31], Microsoft Azure [32], Google Cloud Platform (GCP) [33], and IBM Cloud [34], offer a diverse range of services. These include virtualization services, storage solutions, and networking capabilities.

For instance, cloud providers offer virtual computing instances, commonly referred to as virtual machines (VMs), with varying levels of performance, memory, and storage capacity. These VMs are hosted on servers distributed across different geographic locations, known as regions or availability zones, and are accessible remotely via an internet connection. Additionally, cloud computing platforms have become a popular choice for provisioning network resources, such as individual IP addresses or entire subnets. These resources can be allocated from the cloud provider's pool of public IP addresses and may belong to different regions worldwide.

Cloud providers can offer such services on the basis of different pricing model solutions. Some of them include [35]:

- **On-demand Pricing**. This pricing model, commonly referred to as the pay-as-you-go model, allows customers to pay only for the resources they use. It is ideal for applications that require high availability and cannot tolerate interruptions.

- **Spot pricing.** This is an auction-based system where customers can bid for resources at a lower price than the on-demand rate. However, the provider has not commitment to users and can terminate resources at any time if the spot price exceeds the user's bid. Spot pricing offer a cost-effective option for applications that can handle interruptions.

2.4.1 Cloud Telescope

Due to its characteristics, cloud computing has become a popular platform also for deploying network telescopes. A cloud telescope is a network telescope that is deployed in a cloud environment, leveraging the resources provided by cloud providers to monitor IBR [2] [36].

Figure 2.2 illustrates the vendor-neutral cloud telescope architecture proposed by [36]. The diagram shows the components and nodes that allow a sensor to capture IBR within a specific cloud region. By default, according to RFC1918, the cloud platform assigns the capture sensor, running as a virtualized instance, a private internal IP address. However, to operate effectively within a capturing platform, the sensor requires one or more public IPv4 addresses from the cloud provider's available pool. The Gateway node routes incoming traffic to the sensor, passing it through a Security Group that acts as a firewall. To fully expose the sensor to IBR, the configuration must explicitly allow all incoming traffic through the Security Group.

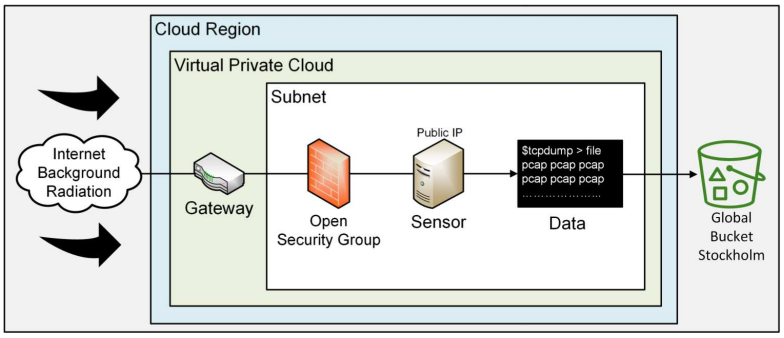


Figure 2.2: Cloud Telescope Architecture [36]

2.5 Data Mining

This section examines a data mining approach to extract insights from large datasets, such as those collected by network telescopes that characterize IBR. By its definition [37], data mining is the process of discovering insightful, interesting, and novel patterns, as well as descriptive, understandable and predictive models from large-scale data. Key data mining tasks include exploratory data analysis, frequent pattern mining, clustering, and classification.

2.5.1 Frequent Pattern Mining

Frequent pattern mining is the process of identifying informative and meaningful patterns within large and complex datasets. These patterns can include sets of co-occurring attribute values, known as *itemsets*, or more intricate structures such as sequences, which account for explicit positional or temporal relationships, and graphs, which capture arbitrary relationships between data points. The primary objective is to uncover hidden trends and behaviors within the data, providing deeper insights into the interactions among attributes and data points. Below, we describe the problem of finding frequent patterns.

- Let $I = \{i_1, i_2, \dots i_m\}$ be the set of all M possible items, and let $T = \{T_1, T_2, \dots T_n\}$ be a set of N transactions, called *database*.

- Each transaction has a unique ID and contains a subset of the items in I.

- The support $\text{supp}(X)$ of a set of item X, where $X \subseteq I$, is defined as the number of transactions in T that contain X.

- An itemset X is considered frequent if its support is greater than or equal to a user defined minimum support threshold.

For instance, let $I = \{A, B, C, D, E\}$ represent the set of all possible items, and consider Table 2.1a as the set of transactions T in the *database*. Table 6.2b lists the frequent itemsets, grouped by support value, that meet an absolute *minsup* threshold of 3.

Transaction ID	Items
1	ABDE
2	BCE
3	ABDE
4	ABCE
5	ABCDE
6	BCD

(a) Transaction database

Support	Itemsets
6	B
5	E, BE
4	A, C, D, AB, AE, BC, BD, ABE
3	AD, CE, DE, ABD, ADE, BCE, BDE, ABDE

(b) Frequent itemsets ($minsup = 3$)

Table 2.1: Transaction database and frequent itemsets

When working with large datasets, the number of possible itemsets can become extremely high, making it computationally expensive to analyze all frequent itemsets. An alternative approach in frequent pattern mining is to focus on identifying a condensed representation of the frequent itemsets, specifically the *maximal frequent itemsets*. A frequent itemset X is considered maximal if it has no frequent superset, meaning there is no other frequent itemset X' such that $X \subset X'$. In the previous example, the frequent itemset BE is not maximal because it has the frequent supersets: ABE, BCE, BDE, and $ABDE$. Conversely, the frequent itemsets $ABDE$ and BCE are maximal because they do not have any frequent itemsets containing them.

2.5.2 Association Rules

An association rule is a logical expression of the form $X \rightarrow Y$, where X and Y are disjoint itemsets ($X, Y \subseteq I$ and $X \cap Y = \emptyset$). A rule indicates that the occurrence of itemset X (antecedent) in a transaction suggests a strong likelihood of the occurrence of itemset Y (consequent) within the same transaction. Important metrics for evaluating association rules include:

- **Support**. The support of a rule represents the proportion of transactions in the database that contain both X and Y. It measures how frequently the rule occurs in the dataset.
- **Confidence**. The confidence of a rule is the conditional probability that a

transaction contains Y given that it contains X. It indicates the reliability of the rule.

- **Lift**. The lift measures the strength of a rule by comparing the observed joint probability of X and Y to their expected joint probability if they were statistically independent. A lift value close to 1 suggests that X and Y occur together as expected under independence. Values significantly greater than 1 indicate a positive association (above expectation), while values significantly less than 1 indicate a negative association (below expectation).

While frequent itemsets represent groups of items that occur with a certain frequency, association rules provide insights into the relationships between these itemsets. They indicate the likelihood that one itemset occurs alongside another within a transaction. Support, Confidence an Lift serves as metrics to evaluate the strength of such relationships.

To generate frequent and high-confidence association rules, the process begins by identifying all frequent itemsets and computing their support values. Given a *database* and a user-defined minimum support threshold (*minsup*), frequent itemset mining aims to find all itemsets whose support values are greater than or equal to *minsup*. After identifying these frequent itemsets, association rule mining derives all strong rules that also meet a user-defined minimum confidence threshold.

For example, consider the frequent itemsets in Table 2.1b, obtained with a *minsup* threshold of 3. We aim to find the association rules that meet a minimum confidence threshold of 0.7. Below are two sample rules and their metric computations. While the first rule meets the minimum confidence threshold of 0.7, the second rule does not.

1. **Rule:** $BC \rightarrow E$
   - **Support:** $sup(BC \rightarrow E) = sup(BCE) = 3$
   - **Confidence:** $conf(BC \rightarrow E) = \frac{sup(BCE)}{sup(BC)} = \frac{3}{4} = 0.75$
2. **Rule:** $B \rightarrow A$
   - **Support:** $sup(B \rightarrow A) = sup(AB) = 4$
   - **Confidence:** $conf(B \rightarrow A) = \frac{sup(AB)}{sup(B)} = \frac{4}{6} \approx 0.67$

Frequent pattern mining and association rule mining are widely used in various domains. The most notable is market basket analysis where they help identifying patterns in customer purchasing behaviour. Another application is network traffic analysis. They are useful for identifying patterns of general network activity, as well as common behaviours of groups of hosts. For example, we can use frequent pattern mining for characterizing scans actors by detecting frequent combinations of targeted ports.

Chapter 3

Exploration of providers for a cloud based telescope

In this chapter, we explore various strategies and solutions for obtaining the essential resources needed to implement a cloud-based network telescope. The key resources include networking components and one or more virtual machines capable of being assigned multiple IP addresses. In Section (2.3.1), we discussed the importance of network telescope size as a key factor for its effectiveness in capturing a significant amount of unsolicited traffic. Based on this, we chose to provision an IP address space equivalent to a /24 subnet (256 public IP addresses). Additionally, we required these IP addresses to be contiguous in order to emulate the setup of a traditional network telescope hosted on our campus IP space. This contiguity also helps avoid potential bias in the collected data, as IPs located at the edges of subnets are often more prone to receiving anomalous or less representative traffic.

We propose two primary approaches: acquiring public IP addresses directly from a cloud provider or leasing them from a third-party organization. In Section (1.3.1), we discussed works that provision resources using *spot* pricing models, which often result in instances being terminated and replaced frequently. Such instability makes it unsuitable for monitoring a consistent set of IP addresses over an extended period. Therefore, our goal is to provision static resources using the *on-demand* pricing model, ensuring they remain consistent over a one-month period.

3.1 Cloud providers

In this section, we analyze the major cloud providers and their solutions for obtaining public IP addresses. We consider the costs associated with provisioning virtual machines and allocating a /24 subnet (256 public IP addresses) to them.

We evaluated various approaches to achieve the objective in the most feasible and cost-effective manner. This included, according to the provider's capabilities, comparing the costs of associating 256 IP addresses to a single virtual machine versus provisioning multiple virtual machines, each with a smaller number of IP addresses. In general, we do not need VMs with high-performance capacities, as we do not require them to perform complex computations. Our main requirement is the ability to associate the predefined set of IP addresses to one or more virtual instances.

For each provider we present different solutions, including the price of the virtual machines and costs related to network resources. Notably, the network capabilities of an instance vary by provider and depend on the instance type. Additionally, due to capability constraints, some provider solutions do not allow the allocation of exactly 256 IP addresses. Instead, they enable the allocation of an approximation of a /24 subnet by provisioning 240 IP addresses. The proposed solutions include details such as the instance type, the number of virtual machines required to obtain the desired number of IP addresses, the hourly cost of each instance (which increases with its size and performance), and the total monthly cost. The monthly cost is calculated based on the usage of all required instances for 730 hours. Then, we provide details about network pricing, according to the specific provider capabilities.

3.1.1 Amazon Web Service (AWS)

Amazon Web Services (AWS) [31] offers a wide range of virtual machines, known as Amazon EC2 instances. Table 3.1 provides a comparison of various AWS instance types and their associated costs for monitoring a /24 subnet. AWS instances can attach multiple network interfaces (NICs), each capable of hosting several IPv4 addresses. However, the limits on the number of NICs and IPs per NIC depend on the instance type. For example, smaller instances like `t4g.nano` or `t4g.micro` support up to 2 NICs with up 2 IPv4 addresses per NIC. In contrast, larger instances such as `m5.4xlarge` and `a1.4xlarge` support up to 8 NICs, each capable of hosting up to 30 IPv4 addresses. Thus, it is not possible to allocate exactly 256 IP addresses, but we can approximate a /24 subnet by provisioning 240 addresses. To do this, one would need to provision a minimum of 60 virtual machines using smaller instance types like `t4g.nano` or `t4g.micro`, or a single virtual machine using larger instance types like `m5.4xlarge` or `a1.4xlarge`. Table 3.1 shows the total price for each solution, demonstrating that acquiring 60 `t4g.nano` instances is more cost-effective than using a single large instance. However, the minimum number of instances required to achieve the goal is very high. This increases the complexity of management and configuration. Therefore, we can consider the `a1.4xlarge` instance as a more feasible option, despite its higher cost.

Instance Type	vCPUs	Memory (GiB)	VMs required	Price/VM-hour	Total VM Price/month (730 h)
t4g.nano	2	0.50	60	$0.0042	$183.96
t4g.micro	2	1.00	60	$0.0084	$367.92
m5.4xlarge	16	64	1	$0.7680	$560.64
a1.4xlarge	16	32	1	$0.4080	$297.84

Table 3.1: Comparison of selected AWS instance types

In AWS, public IP addresses are called *elastic IPs*. These are static IPv4 addresses designed for dynamic cloud computing, allowing users to remap them to different instances in case of failure. Elastic IPs are allocated from AWS's global address pool, which does not guarantee contiguous addresses and therefore does not support prefix allocation. The cost of Elastic IPs is $0.005 per hour. For 240 addresses, the cost is calculated as $0.005 \times 240 = \$1.2$ per hour. Over 730 hours in a month, this results in a total cost of $876, to add to the cost of the virtual machines.

3.1.2 Google Cloud

Google Cloud Platform [33] proposes a diverse range of virtual machine types, referred as Google Compute Engine instances. It allows assigning only one public IPv4 address per network interface, and the maximum number of virtual network interfaces (vNICs) per VM depends on the number of virtual CPUs (vCPUs). For example, an `e2-highcpu-8` instance, equipped with 8 vCPUs, supports up to 8 network interfaces, allowing for the allocation of up to 8 public IP addresses per instance. To allocate 256 public IP addresses, a minimum of 32 such instances would be required. Table 3.2 compares the costs of different instance configurations, demonstrating that using 32 instances of type `e2-highcpu-8` is more cost-effective than provisioning 128 instances of type `e2-highcpu-2` or 256 instances of type `c4a-high-cpu-1`.

Instance Type	vCPUs	Memory (GiB)	VMs required	Price/VM-hour	Total VM Price/month (730 h)
c4a-high-cpu-1	1	2	256	$0.03788	$7079.01
e2-highcpu-2	2	2	128	$0.04950	$4625.28
e2-highcpu-8	8	8	32	$0.19790	$4622.94

Table 3.2: Comparison of selected Google instance types

In Google Cloud, the cost of public IP addresses is $0.005 per hour. Therefore, for 256 addresses, the cost is $0.005 \times 256 = \$1.28$ per hour, which multiplied by 730 hours in a month results in $934.40 per month. As for AWS, Google Cloud does not support the allocation of contiguous blocks of IP addresses.

3.1.3 Microsoft Azure

Microsoft Azure [32] allows associating multiple public IP addresses to a single virtual machine (VM) and multiple addresses to a single network interface (NIC). Azure defines a theoretical limit of 256 public IP addresses per VM, and also 256 public IP addresses per NIC. For this reason, it is sufficient to consider a single VM with minimal computational capabilities to achieve the goal of monitoring a /24 subnet. However, smaller instances might not support such a high number of IP addresses. Thus, we applied the same approach as with AWS and Google Cloud, comparing the costs of provisioning a single large instance versus multiple smaller instances. Table 3.3 compares the costs of different solutions. It shows how using a single general purpose instance of type `D2pldsv5` is more cost-effective than provisioning 256 instances of type `B1ls`, which are the smallest and cheapest instances available in Azure.

Instance Type	Cores	Memory (GiB)	VMs required	Price/VM-hour	Total VM Price/month (730 h)
B1ls	1	0.5	256	$0.0048	$897.02
D2plds v5	2	4.0	1	$0.0711	$51.90

Table 3.3: Comparison of selected Azure instance types

In Azure, the cost of a public IP address is $0.005 per hour. For 256 addresses, this results in a cost of $0.005 $\times$ 256 = $1.28 per hour, which over 730 hours in a month amounts to $934.40 per month.

Moreover, Azure supports the allocation of contiguous blocks of IP addresses. The maximum size of an allocable subnet is /28, containing up to 16 IP addresses, each costing $0.006 per hour. To obtain 256 IP addresses belonging to an IP prefix, the cost is $0.006 $\times$ 256 = $1.536 per hour, which over 730 hours in a month totals $1,121.28 per month.

3.2 IP Leasing

Acquiring an address space directly from a cloud provider can be expensive, especially when requiring a large number of public IP addresses such as a /24 subnet. An alternative solution is IP leasing. This approach involves renting IP addresses from a third-party organization that provides the addresses without offering connectivity. To make these IP addresses reachable on the Internet, they must be advertised via BGP (Border Gateway Protocol). The *lesse* can either advertise the address space themselves, or it can utilize BYOIP (Bring Your Own IP) services offered by cloud providers that will act as *originator* of the advertised prefix. Figure 3.1 illustrates the key entities involved in the IP leasing business model, detailing the entire process from the IP addresses acquisition to the advertisement process.

- **IP Holder**: The entity that owns and manages the IP address space. This could be an organization allocated IP addresses by a Regional Internet Registry (RIR) or an entity that acquired them through other means.
- **Lessee**: The entity that temporarily rents the IP address space from the IP holder.
- **Facilitator**: An intermediary that, facilitates the transaction between the IP holder and the lessee. However, the leasing process does not always require a facilitator.
- **Originator**: The entity responsible for advertising the leased IP address space in BGP. This is typically the entity behind the origin AS (Autonomous System) of the advertised prefix. The originator may also be the lessee if they have their own routing infrastructure.

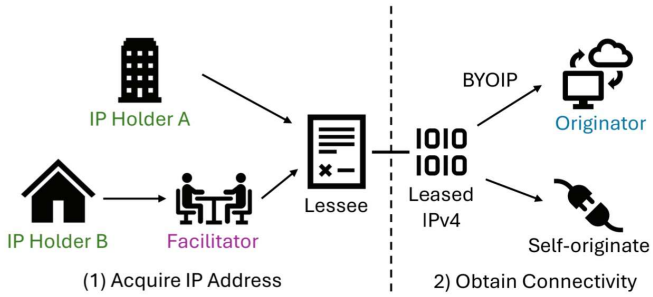


Figure 3.1: IP Leasing Business Model [38]

One of the most notable facilitators in the IP leasing market is IPXO [39], a leading IP broker that specializes in IP leasing services. IPXO ranks among the top three facilitators in the RIPE, ARIN, and APNIC regions. Table 3.4 provides an overview of the costs associated with leasing IP address blocks from IPXO.

Subnet Mask	Monthly Price
/24 (256 IPs)	$91.86
/23 (512 IPs)	$186.29

Table 3.4: IP leasing costs by subnet mask

After leasing the subnet, IPXO provides two options for advertising the IP addresses: *self-originate* and *BYOIP* service. In both cases, advertisement consists into announcing the leased IP prefixes via the Border Gateway Protocol (BGP),

making them globally reachable on the Internet. In the first option, the lessee advertises the leased IP addresses using their own Autonomous System Number (ASN) or that of their Internet Service Provider (ISP). In the second option, after verifying ownership of the IP addresses, the lessee delegates the advertisement process to the cloud provider. The cloud provider then announces the leased IP addresses through its own ASN using Bring Your Own IP (BYOIP) services. AWS, Google Cloud, and Microsoft Azure propose a multi-step BYOIP process for bringing IP address space within their infrastructure.

3.3 Final choice

This section summarizes the costs of various solutions and identifies the most cost-effective and feasible option for implementing a cloud telescope.

Although IP leasing can be cost-effective, we excluded it for several reasons. First, the *self-originate* option requires the lessee to perform BGP announcements and associate the leased subnet with their own ASN, which increases the complexity of both setup and management. Moreover, in this configuration the leased address space appears as part of our organization rather than the cloud provider's infrastructure, which is inconsistent with our objectives. The *BYOIP* option mitigates this issue by allowing the cloud provider to advertise the leased subnet under its own ASN, thereby integrating it into its infrastructure. While this approach better aligns with our objectives, our study specifically focuses on analyzing traffic directed to IP ranges already belonging to the cloud provider's pools, since prior usage of those addresses may influence observed activity. Ultimately, despite their competitive pricing, both leasing options introduce complexity and administrative overhead.

For these reasons, the focus shifted to solutions involving the acquisition of one or more virtual machines from a cloud provider, along with a set of 256 public IP addresses. Table 3.5 provides a detailed comparison of costs across the three main cloud providers, including expenses for virtual machines, public IP addresses, and additional costs such as outbound traffic and storage. The total cost accounts for potential price fluctuations based on the region selected for the VM, along with additional hours required for configuring the platform before making it operational.

Based on the cost analysis, the two most cost-effective solutions are AWS and Microsoft Azure. However, AWS does not support allocating multiple public IP addresses within the same subnet, which conflicts with the objectives of this thesis. In contrast, Azure allows the allocation of contiguous blocks of IP addresses, with a maximum block size of /28 (16 addresses per block). Therefore, we selected Microsoft Azure and allocated 16 separate /28 subnets to achieve the required address space of 256 IP addresses.

Provider	Virtual Machine	Number of VMs	VM Cost ($/h)	IP Cost ($/h)	VM + IPs for 730h[3] ($)	Extra ($)	Total ($)
Amazon Web Service	a1.4xlarge	1	0.408	0.005	1174	1 TB Outbound Traffic + $92 Disk + $4	1320
Google Cloud Platform	e2-highcpu-8	32	0.197	0.005	5557	1 TB Outbound Traffic + $122	5700
Microsoft Azure	D2plds v5	1	0.077	0.006	1177	1 TB Outbound Traffic + $75	1300

Table 3.5: Cost comparison of cloud providers for VM and IP allocation

3.4 Summary of findings

This chapter explored various strategies for acquiring the resources needed to implement a cloud-based network telescope. The primary objective was to obtain a /24 subnet (256 public IP addresses) and the virtual machines (VMs) required to host these addresses. Table 3.6 provides a summary of the solutions considered, evaluating them based on cost-effectiveness, support for contiguous IP address allocation, and ease of setup. While the IP leasing option proved to be cost-effective, we excluded it due to the significant administrative overhead and the complexity involved in setup and ongoing management. Among the cloud providers, Microsoft Azure emerged as the most suitable choice, offering a balance of affordability, support for contiguous IP address allocation, and simplicity in deployment.

Solution	Cost Effectiveness	Address Contiguity	Ease of Setup
IP Leasing	✓	✓	✗
Amazon Web Service (AWS)	✓	✗	✓
Google Cloud (GCP)	✗	✗	✗
Microsoft Azure	✓	✓	✓

Table 3.6: Summary of solutions

[3]For AWS and Google Cloud, the cost includes the acquisition of 240 public IP addresses. For Azure, the cost includes 256 public IP addresses.

Chapter 4

Azure testing and implementation

This chapter details the implementation process of a network telescope using Microsoft Azure's cloud infrastructure. It begins with an overview of Azure's network architecture, followed by tests assessing whether the cloud infrastructure imposes any limitations, such as undeclared network components that filter (e.g., firewalls) or modify (e.g., NAT devices) incoming traffic. Finally, we outline the implementation steps, including IP space allocation and deployment of interactive services such as Layer 4 responders and honeypots.

4.1 Azure network architecture overview

Before conducting the tests on the Azure cloud infrastructure, it is important to first examine the key components of Azure's network architecture. In Chapter 2, we described a vendor-neutral cloud telescope architecture (2.4.1). In this section, we define Azure's architecture based on its terminology and components.

Figure 4.1 provides a high-level overview of the architecture [32], illustrating the workflow of incoming traffic directed to a Virtual Machine (VM). The VM resides in a virtual subnet within a *Virtual Network* (VNet). It is connected to a *Network Interface* (NIC), which facilitates communication with other resources in the virtual network and the internet. Each NIC is assigned a default IP configuration, which includes a private IP address and, optionally, a public IP address. The public IP address is essential for enabling communication with the VM from the internet.

Incoming traffic first passes through a *Network Address Translation* (NAT)[1]

[1]It refers to Azure's default NAT functionality, not the NAT Gateway service.

device, which handles the mapping between public and private IP addresses. It then reaches a *Network Security Group* (NSG), which acts as a stateful packet filtering firewall. The NSG allows or denies inbound and outbound network traffic based on user defined security rules. It can be associated either to a subnet or a NIC.

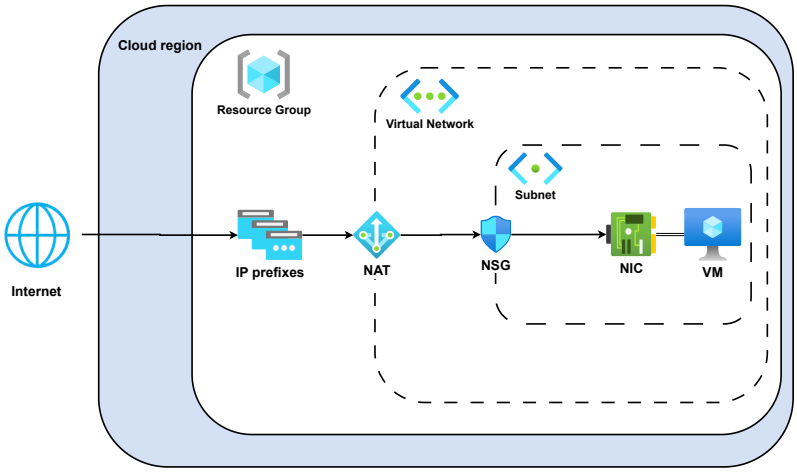


Figure 4.1: Azure Network Architecture

4.2 Does the cloud bring limitations for network telescope?

In this section, we investigate if the cloud environment introduces some constraint or limitation for the deployment of a network telescope. Specifically we aim to determine whether the cloud infrastructure comprises of undeclared firewalls or NAT devices that brings filtering or alteration of incoming traffic. Our goal is to receive traffic from the entire IPv4 address space on all ports. To evaluate this, we conducted a series of tests using an Azure Virtual Machine (VM), specifically a Standard D2s v3 instance (2 vCPUs, 8 GiB memory), chosen for its cost-effectiveness. We configured it with a single public IP address.

To conduct tests effectively, it is essential to establish a basic networking setup for the Azure VM. We have observed in Section (2.4.1) and (4.1) that the VM, where the network telescope sensor is located, resides within a Virtual Private Cloud region, where incoming traffic flows through a Virtual Gateway and a Network Security Group (NSG). The NSG acts as a virtual firewall, filtering incoming and outgoing traffic based on user-defined rules. Table 4.1 provides an overview of the NSG configuration set up directly in the Azure portal interface. It highlights

the custom rules we implemented alongside the default rules provided by Azure. The rules *AllowAnyCustomAnyInbound* and *AllowAnyCustomAnyOutbound* permit all incoming and outgoing traffic, respectively, and have an higher priority (lower number) than the default rules *DenyAllInbound* and *DenyAllOutbound*, which Microsoft sets to block all traffic by default. Notably, we use port 22 (SSH) for connecting to the VM to perform tests and management operations.

Priority	Name	Port	Protocol	Source	Destination	Action
301	**AllowAnyCustomAnyInbound**	**Any**	**Any**	**Any**	**Any**	**Allow**
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny
311	**AllowAnyCustomAnyOutbound**	**Any**	**Any**	**Any**	**Any**	**Allow**
65000	AllowVnetOutBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowInternetOutBound	Any	Any	Any	Internet	Allow
65500	DenyAllOutBound	Any	Any	Any	Any	Deny

Table 4.1: Network Security Group (NSG) setup

4.2.1 Testing

In this section, we describe the tests conducted to evaluate whether the cloud infrastructure includes any undeclared network components, such as firewalls or NAT devices, that could filter or modify unsolicited traffic. First, we performed tests to verify the effectiveness of basic *nmap* scans, and whether all packets successfully reach the target machine. Next, we investigated whether the cloud infrastructure performs payload inspection by sending packets containing either benign or known malicious payloads. Finally, we perform a capture of unsolicited traffic to verify whether the telescope can effectively collect scanning activity from the entire IBR.

4.2.2 Nmap test

The primary goal of a network telescope is to capture scanning activities. To verify whether the telescope can detect traces of scanning activities, such as sequences of SYN packets, we conducted tests using the widely used scanning tool *nmap* [8]. These tests aimed to ensure that the scan results accurately reflect the current state of the virtual machine i.e., ports status and host availability, and that all packets sent from the scanning source successfully reach the target machine.

The tests involved scanning the first 1000 ports from an external host while capturing the incoming traffic on the target machine using `tcpdump` [27]. We performed two scans: a basic scan where all ports were closed except for port 22 (used for SSH connections), and a second scan where port 8080 was opened to host a simple web service. Listings 4.1 and 4.2 present the results of these scans, which

correctly identified the host as active and accurately reflected the state of the ports on the target machine. All SYN packets successfully reached the destination, and closed ports responded with RST packets, confirming their status. These results demonstrate that performing a basic *nmap* on a cloud instance is effective and accurately reflects the VM's current state.

```
Nmap scan report for ${VM_PUBLIC_IP_ADDRESS}
Host is up (0.11s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh

Nmap done: 1 IP address (1 host up) scanned in 3.51 seconds
```

Listing 4.1: Nmap basic scan

```
Nmap scan report for ${VM_PUBLIC_IP_ADDRESS}
Host is up (0.11s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
8080/tcp  open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 3.04 seconds
```

Listing 4.2: Open service scan

To further validate the test, we performed a detailed inspection of packet traces captured on both source and target machine after the basic *nmap* scan. The source machine sent a total of 2009 packets, while the target machine received 2007 packets. Figures 4.2a and 4.2b explain the slight difference in the number of packets observed. Figure 4.2a highlights two packets (no. 20 and no. 21) that did not reach the target machine: an ACK packet directed to port 80 and an ICMP timestamp request. Although a basic *nmap* scan is designed to send only SYN packets, these two packets are default behaviors of *nmap*, used to determine if the target host is alive when no specific host discovery options are provided [8].

No.	Time	Source	Destination	Protocol	Length	Info
18	4.311306	192.168.0.68	20.169.159.227	ICMP	42	Echo (ping) request id=0x46b4, seq=0/0, ttl=46 (reply in 22)
20	4.312871	192.168.0.68	20.169.159.227	TCP	54	36021 → 80 [ACK] Seq=1 Ack=1 Win=1024 Len=0
21	4.313582	192.168.0.68	20.169.159.227	ICMP	54	Timestamp request id=0x4e68, seq=0/0, ttl=46
22	4.427449	20.169.159.227	192.168.0.68	ICMP	60	Echo (ping) reply id=0x46b4, seq=0/0, ttl=46 (request in 18)

(a) Source machine packet capture

No.	Time	Source	Destination	Protocol	Length	Info
8	7.866435	101.56.150.161	10.0.0.5	ICMP	60	Echo (ping) request id=0x0751, seq=0/0, ttl=27 (reply in 9)
9	7.866478	10.0.0.5	101.56.150.161	ICMP	42	Echo (ping) reply id=0x0751, seq=0/0, ttl=64 (request in 8)

(b) Target machine packet capture

Figure 4.2: Capture comparison between source and target machines

In Section (4.1), we explained that the NSG functions as a *stateful* firewall, meaning it tracks the state of flows and allows or denies packets based on their connection state. In this case, the NSG dropped the ACK packet because it was not part of any established handshake. On the other hand, Azure platform dropped

the ICMP timestamp request packet due to the ICMP Timestamp Request Remote Date Disclosure vulnerability (CVE-1999-0524) [40]. This vulnerability arises when an attacker sends an ICMP timestamp request to a target system, which then responds with its system's timestamp information, potentially exposing the exact time of the target system.

This test also raised an important considerations: the Azure Virtual Machine resides in a Virtual Private Network, where it is assigned a private IP address within the private subnet (Figure 4.1). The `tcpdump` operation captures traffic arriving at the VM's network interface, and the address 10.0.0.5 corresponds to the VM's private IP address. As a result, the public IP address assigned to the VM is not visible in the capture.

4.2.3 Sending payloads

In the previous section, we demonstrated that the cloud infrastructure does not impose significant limitations on basic *nmap* scans, as all SYN packets successfully reached the target destination, and the scan results accurately reflected the current state of the ports.

In this section, we perform a capture comparison by sending packets with specific payloads to the VM to determine whether the cloud infrastructure modifies any application-level messages during transmission. We opened a service on the target machine on port 8080 and sent a payload to the target machine using the following command: `echo $TEXTURE_PAYLOAD | ncat $TARGET_IP_ADDRESS $TARGET_PORT`. Figure 4.3 illustrates the handshake that source and destination completed to transmit the packet containing the payload. Notably, it shows how the general packet length (81 bytes) and the TCP payload length (27 bytes) remain consistent between the source and target machine.

No.	Time	Source	Destination	Protocol	Length	Info
143	13.609490	172.25.51.9	20.169.159.227	TCP	66	26427 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
144	13.715294	20.169.159.227	172.25.51.9	TCP	66	8080 → 26427 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1440 SACK_PERM WS=128
145	13.715446	172.25.51.9	20.169.159.227	TCP	54	26427 → 8080 [ACK] Seq=1 Ack=1 Win=65280 Len=0
147	13.847097	172.25.51.9	20.169.159.227	TCP	81	26427 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=27
148	13.847219	172.25.51.9	20.169.159.227	TCP	54	26427 → 8080 [FIN, ACK] Seq=28 Ack=1 Win=65280 Len=0
152	13.944974	20.169.159.227	172.25.51.9	TCP	54	8080 → 26427 [ACK] Seq=1 Ack=28 Win=64256 Len=0
153	13.944974	20.169.159.227	172.25.51.9	TCP	54	8080 → 26427 [FIN, ACK] Seq=1 Ack=29 Win=64256 Len=0
156	13.945129	172.25.51.9	20.169.159.227	TCP	54	26427 → 8080 [ACK] Seq=29 Ack=2 Win=65280 Len=0

(a) Source machine packet capture

No.	Time	Source	Destination	Protocol	Length	Info
9	12.729597	130.192.232.232	10.0.0.5	TCP	66	10400 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
10	12.729655	10.0.0.5	130.192.232.2…	TCP	66	8080 → 10400 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
11	12.835033	130.192.232.232	10.0.0.5	TCP	60	10400 → 8080 [ACK] Seq=1 Ack=1 Win=65280 Len=0
12	12.964806	130.192.232.232	10.0.0.5	TCP	81	10400 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=27
13	12.964806	130.192.232.232	10.0.0.5	TCP	60	10400 → 8080 [FIN, ACK] Seq=28 Ack=1 Win=65280 Len=0
14	12.964864	10.0.0.5	130.192.232.2…	TCP	54	8080 → 10400 [ACK] Seq=1 Ack=28 Win=64256 Len=0
15	12.964969	10.0.0.5	130.192.232.2…	TCP	54	8080 → 10400 [FIN, ACK] Seq=1 Ack=29 Win=64256 Len=0
16	13.063921	130.192.232.232	10.0.0.5	TCP	60	10400 → 8080 [ACK] Seq=29 Ack=2 Win=65280 Len=0

(b) Target machine packet capture

Figure 4.3: Capture comparison between source and target machines with payload

However, this test also revealed another important consideration. Figure 4.3 shows that the source ports of packets differed between the source and destination machines because the source machine was behind a NAT that modified the source port before the packets reached the destination. In a separate experiment, we observed that packets sent from a source machine with a public IP address retained their original source ports. This behavior indicates that the observed source port changes are caused by the NAT device in front of the source machine, rather than modifications by the cloud infrastructure.

We also tested whether the cloud infrastructure inspects packets for suspicious content. To do this, we attempted to send known malicious payloads, such as:

- Malformed HTTP requests: it is a common technique used by attackers to exploit vulnerabilities in web servers. We sent a request containing a null byte (%00) to the target machine on port 80.

```
echo -ne "GET /%00 HTTP/1.1\r\nHost: $TARGET_IP\r\n\r\n" | nc -n -v -w 5 $TARGET_IP 80
```

- Malformed SSH packet: this is a crafted binary packet used to elicit parsing errors, fingerprint behavior, or other responses from SSH servers. it is composed of a sequence of null bytes followed by a specific byte sequence.

```
echo -ne "\x00\x00\x00\x05\x00\x00\x00\x00\x00" | nc -n -v -w 5 $TARGET_IP 22
```

- Telnet option negotiation: it is a sequence of bytes used in the Telnet protocol to negotiate options between client and server [41]. Specifically, we sent three sequence of bytes (`0xFF 0xFD <opt>`) determining the request for three options code: `x18` (Terminal Type), `x20` (Terminal speed) and `x23` (X Display location). These commands are useful to elicit characteristics of the target system.

```
echo -ne "\xff\xfd\x18\xff\xfd\x20\xff\xfd\x23" | nc -n -v -w 5 $TARGET_IP 23
```

In all cases, the payloads arrived unaltered at the destination, confirming that the cloud infrastructure does not perform particular application-level inspections during transmission.

4.2.4 Collecting unsolicited traffic

In the previous sections, we demonstrated that scanning activities can accurately identify closed and open ports on our VM, and that almost all packets successfully reach the target machine without alteration or filtering by the cloud infrastructure. This section focuses on capturing incoming traffic using `tcpdump` [27] to verify whether the internet scanners reach our VM.

We performed a 7-hour capture, recording all traffic directed to the VM, collecting 64 MB of data across 231,350 packets. Within the captured traffic, we can clearly identify examples of scanning activity. Figures 4.4a and 4.4b illustrate two of them. The first figure shows a standard TCP scan, where multiple scanners send SYN packets to various ports and receive RST responses from our VM for closed ports. The second figure depicts a stealth scan, where a single scanner avoids completing the TCP handshake by sending a SYN packet followed by an RST packet. However, since all ports are closed, it consistently receives RST responses from our VM.

No.	Time	Source	Destination	Protocol	Length	Info
2388	67.771756	162.216.150.128	10.0.0.4	TCP	60	53829 → 64227 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
2389	67.771811	10.0.0.4	162.216.150.128	TCP	54	64227 → 53829 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
2428	72.963866	162.216.149.43	10.0.0.4	TCP	60	55894 → 9405 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
2429	72.963924	10.0.0.4	162.216.149.43	TCP	54	9405 → 55894 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
2431	74.290898	35.203.211.231	10.0.0.4	TCP	60	53992 → 23917 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
2432	74.290951	10.0.0.4	35.203.211.231	TCP	54	23917 → 53992 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
2467	74.946110	147.185.133.12	10.0.0.4	TCP	60	50502 → 33013 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
2468	74.946164	10.0.0.4	147.185.133.12	TCP	54	33013 → 50502 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
2475	79.039422	35.203.211.233	10.0.0.4	TCP	60	54480 → 47342 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
2476	79.039475	10.0.0.4	35.203.211.233	TCP	54	47342 → 54480 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
2477	79.728377	35.203.211.22	10.0.0.4	TCP	60	49777 → 9874 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
2478	79.728420	10.0.0.4	35.203.211.22	TCP	54	9874 → 49777 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

(a) TCP scan

No.	Time	Source	Destination	Protocol	Length	Info
15171	1267.859531	92.63.197.66	10.0.0.4	TCP	60	8080 → 65456 [SYN] Seq=0 Win=1024 Len=0
15172	1267.859583	10.0.0.4	92.63.197.66	TCP	54	65456 → 8080 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15173	1267.958489	92.63.197.66	10.0.0.4	TCP	60	8080 → 65456 [RST] Seq=1 Win=1200 Len=0
22030	2112.706091	92.63.197.66	10.0.0.4	TCP	60	8080 → 65390 [SYN] Seq=0 Win=1024 Len=0
22031	2112.706143	10.0.0.4	92.63.197.66	TCP	54	65390 → 8080 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
22032	2112.799276	92.63.197.66	10.0.0.4	TCP	60	8080 → 65390 [RST] Seq=1 Win=1200 Len=0
31557	2841.903780	92.63.197.66	10.0.0.4	TCP	60	8080 → 65099 [SYN] Seq=0 Win=1024 Len=0
31558	2841.903836	10.0.0.4	92.63.197.66	TCP	54	65099 → 8080 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
31559	2842.004924	92.63.197.66	10.0.0.4	TCP	60	8080 → 65099 [RST] Seq=1 Win=1200 Len=0
31768	2862.738829	92.63.197.66	10.0.0.4	TCP	60	8080 → 65421 [SYN] Seq=0 Win=1024 Len=0
31769	2862.738875	10.0.0.4	92.63.197.66	TCP	54	65421 → 8080 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
31770	2862.835487	92.63.197.66	10.0.0.4	TCP	60	8080 → 65421 [RST] Seq=1 Win=1200 Len=0
37381	3533.188213	92.63.197.66	10.0.0.4	TCP	60	8080 → 65369 [SYN] Seq=0 Win=1024 Len=0
37382	3533.188256	10.0.0.4	92.63.197.66	TCP	54	65369 → 8080 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
37383	3533.287954	92.63.197.66	10.0.0.4	TCP	60	8080 → 65369 [RST] Seq=1 Win=1200 Len=0
53672	5738.649908	92.63.197.66	10.0.0.4	TCP	60	8080 → 65455 [SYN] Seq=0 Win=1024 Len=0
53673	5738.649951	10.0.0.4	92.63.197.66	TCP	54	65455 → 8080 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
53674	5738.753082	92.63.197.66	10.0.0.4	TCP	60	8080 → 65455 [RST] Seq=1 Win=1200 Len=0
66183	7394.075424	92.63.197.66	10.0.0.4	TCP	60	8080 → 65092 [SYN] Seq=0 Win=1024 Len=0
66184	7394.075470	10.0.0.4	92.63.197.66	TCP	54	65092 → 8080 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
66185	7394.176192	92.63.197.66	10.0.0.4	TCP	60	8080 → 65092 [RST] Seq=1 Win=1200 Len=0
103995	11265.188979	92.63.197.66	10.0.0.4	TCP	60	8080 → 65067 [SYN] Seq=0 Win=1024 Len=0
103996	11265.189033	10.0.0.4	92.63.197.66	TCP	54	65067 → 8080 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
103997	11265.290995	92.63.197.66	10.0.0.4	TCP	60	8080 → 65067 [RST] Seq=1 Win=1200 Len=0
125162	13772.858380	92.63.197.66	10.0.0.4	TCP	60	8080 → 65091 [SYN] Seq=0 Win=1024 Len=0
125163	13772.858429	10.0.0.4	92.63.197.66	TCP	54	65091 → 8080 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
125164	13772.962240	92.63.197.66	10.0.0.4	TCP	60	8080 → 65091 [RST] Seq=1 Win=1200 Len=0

(b) Stealth scan

Figure 4.4: Internet scanning capture

4.2.5 Final considerations of test results

The tests conducted revealed that the cloud infrastructure does not pose a significant obstacle to collect unsolicited traffic using cloud-based network telescope, as the majority of the traffic reaching the Virtual Machine remains unaltered and unfiltered. However, we identified a few exceptions:

- The NSG, functioning as a stateful firewall, drops packets that do not belong to an established connection (e.g., ACK packets not associated with a prior

SYN). This behavior is expected and does not pose a significant limitation for a network telescope, as most scanning activities involve initiating connections.

- The cloud infrastructure drops certain types of packets, such as ICMP timestamp requests, due to known vulnerabilities that can be exploited using these packets. While this may limit the capture of specific traffic types, it does not significantly affect the overall observable scanning activities, as such packets represent only a negligible fraction of the total traffic.

- The Virtual Machine is located in a private subnet, and the public IP address is mapped to a private IP address. As a result, the public IP address is not visible in packet captures taken at the VM's network interface. However, since each public IP address is mapped to a corresponding private IP address, we can maintain a record of the 1:1 association during the public IP configuration phase.

- When the source machine is located behind a NAT device, the source port of packets always change upon reaching the destination machine. This behavior is not caused by the cloud infrastructure but rather by the NAT device in front of the source machine. While this may affect certain types of analysis, it cannot be considered a limitation of the cloud infrastructure itself, as it is strictly related to the source network configuration.

4.3 Setup and implementation

In this section, we outline the steps taken to configure the Azure Virtual Machine with a specific set of IP addresses to function as a network telescope, and detail the deployment of interactive services on specific segments of the allocated IP space.

4.3.1 Network setup

In Section (4.2), we said that we make use of port 22 for SSH connection and for management purposes. However, we want to ensure that management traffic remains separated from darknet traffic during data capture. Initially, we considered using two NICs: one dedicated to management traffic with its own IP address and the other for the network telescope. After some experiments, we determined that the primary NIC always functions as default gateway, requiring us to listen on both interfaces. To simplify the collection process, we opted to use a single NIC assigning a dedicated IP address for management purposes, while the remaining IP addresses allocated for the network telescope.

IP address space allocation

Creating an IP address can be done manually through the Azure portal interface. However, when managing a large number of IP addresses, the process of allocating and associating them with different IP prefixes becomes time-consuming and error-prone. To avoid this labor-intensive manual task, we developed a script to automate the entire process of IP address space allocation. First, we deploy the desired VM along with its associated NIC via the Azure portal. Then, we execute the script. Algorithm 1 presents the pseudocode for the script, which performs two main tasks: it first creates the required set of IP prefixes, and then allocates the desired number of IP addresses from these prefixes.

In our implementation, we allocated 256 IP addresses across 16 prefixes, each with a size of /28. We used the default IP configuration with the private address 10.1.0.4 for management purposes. For the remaining IP addresses, we associated the 16 public IP addresses of each prefix with a corresponding set of private IP addresses in the range 10.1.p.1 to 10.1.p.16, where p represents the prefix number (ranging from 1 to 16). Table 4.2a shows a sample of the allocated public IP addresses and their corresponding private IP addresses. For example, the IP addresses in the prefix 48.217.46.224/16 map to the private IP prefix 10.1.0.1/16. This mapping directly addresses the limitation discussed in Section (4.2), where the public IP address does not appear in the packet capture.

Algorithm 1 IP Prefix Creation and Address Assignment

Input: `TOTAL_PREFIXES` $\leftarrow 16$, `PREFIX_SIZE` $\leftarrow 28$

Create IP Prefixes:
for $p \leftarrow 1$ **to** `TOTAL_PREFIXES` **do**
    `PREFIX_NAME` $\leftarrow$ "prefix-" $+ p$
    Create public IP prefix `PREFIX_NAME` with size /`PREFIX_SIZE`
end for

Allocate Private/Public IPs:
for $p \leftarrow 1$ **to** `TOTAL_PREFIXES` **do**
    for $i \leftarrow 1$ **to** 16 **do** ▷ each /28 has 16 addresses
        `PRIVATE_IP` $\leftarrow$ 10.1.p.i
        `PUBLIC_IP_NAME` $\leftarrow$ "myPublicIP-" $+ p +$ "-" $+ i$
        `PUBLIC_IP` $\leftarrow$ allocate next public IP from the prefix p
        Bind `PRIVATE_IP` to `PUBLIC_IP` via `PUBLIC_IP_NAME`
    end for
end for

Output: Mapping $\langle$`PRIVATE_IP`, `PUBLIC_IP`$\rangle$ for all allocated addresses

This iterative process ensured that the 16 contiguous public IP addresses within each prefix were consistently mapped to their corresponding private IP addresses in the specified range. Table 4.2b summarizes the allocated IP prefixes. We carefully chose prefixes to be as close as possible to one another, effectively simulating the

allocation of a /24 subnet of contiguous IP addresses.

Name	Private IP	Public IP
myPublicIP-1-1	10.1.1.1	48.217.46.233
myPublicIP-1-2	10.1.1.2	48.217.46.227
myPublicIP-1-3	10.1.1.3	48.217.46.237
myPublicIP-1-4	10.1.1.4	48.217.46.231
myPublicIP-1-5	10.1.1.5	48.217.46.224
myPublicIP-1-6	10.1.1.6	48.217.46.228
myPublicIP-1-7	10.1.1.7	48.217.46.229
myPublicIP-1-8	10.1.1.8	48.217.46.238
myPublicIP-1-9	10.1.1.9	48.217.46.235
myPublicIP-1-10	10.1.1.10	48.217.46.225
myPublicIP-1-11	10.1.1.11	48.217.46.230
myPublicIP-1-12	10.1.1.12	48.217.46.234
myPublicIP-1-13	10.1.1.13	48.217.46.239
myPublicIP-1-14	10.1.1.14	48.217.46.232
myPublicIP-1-15	10.1.1.15	48.217.46.226
myPublicIP-1-16	10.1.1.16	48.217.46.236

(a) Private/Public IP Mapping (Prefix 1)

Prefix	Network
1	48.217.46.224/28
2	48.217.46.240/28
3	48.217.46.160/28
4	48.217.46.192/28
5	48.217.46.112/28
6	48.217.46.64/28
7	172.210.53.176/28
8	48.217.46.96/28
9	48.217.46.144/28
10	172.172.155.64/28
11	48.217.47.0/28
12	74.235.93.32/28
13	48.217.46.80/28
14	48.217.46.128/28
15	48.217.46.208/28
16	48.217.46.176/28

(b) IP Prefix Allocation

Table 4.2: Private/Public IP mapping and prefix allocation

4.3.2 Interactive deployments

Monitoring internet traffic collected by network telescopes is typically limited to observing connection attempts, such as TCP SYN packets or ICMP echo requests. However, by deploying interactive services on specific subnets, we respond to such connection attempts, allowing us to observe the subsequent actions of the scanner. This interaction enables us to gain deeper insights into the behavior of internet scanners and potentially attract more traffic. In Section (2.3.1), we discussed the different variants of a network telescope and their corresponding levels of interactivity. Here, we detail the deployment types used for specific subnets across the allocated IP space.

- **Darknet**. This deployment is a simple network telescope, where all addresses remain inactive, with no responses provided to incoming traffic.
- **Layer-4 responder**. In this deployment type, we respond to incoming traffic completing the TCP handshake and sending a TCP RST packet to terminate the connection either (i) after the first packet sent by the sender or (ii) after an idle timeout. This service is active on all the range of ports (1-65535).
- **Honeypot**. In this deployment, we emulate the presence of a Cowrie honeypot [42], and a nginx web server [43]. The former is a medium-interaction SSH and Telnet honeypot and is active on port 22. The latter is a low-interaction HTTP and HTTPS honeypot, active on ports 80 and 443.

In addition to such deployments, we propose some variants. Specifically, we created duplicate versions where we announce the subnet via DNS and public certificate emissions. The objective is to assess whether these features enhance the visibility of the subnets, thereby attracting more traffic.

Table 4.3 provides an overview of different subnets and their corresponding deployment type. In Section (4.3.1) we defined the set of allocated prefixes for the Azure cloud-based network telescope. Table 4.3a presents such allocated prefixes, along with their corresponding deployment types. We replicated the same configuration on our campus IP space, except for the variants with DNS announcements and public certificate emissions. The campus network telescope utilized a /24 subnet within the range 130.192.166.0/24. Table 4.3b outlines the deployment assignments for different segments of the campus subnet.

Prefix ID	**Network**	**Deployment**
7	172.210.53.176/28	Darknet_DNS_Cert
10	172.172.155.64/28	L4Responder_Pure
11	48.217.47.0/28	L4Responder_DNS_Cert
12	74.235.93.32/28	Honeypot_DNS_Cert
Others	48.217.46.x/28	Darknet_Pure

(a) Cloud Deployment Assignment

Network	**Deployment**
From 130.192.166.101 to 130.192.166.116	Honeypot
130.192.166.16/28	L4Responder
All other IPs	Darknet

(b) Campus Deployment Assignment

Table 4.3: Deployment Assignments for Cloud and Campus IP spaces

Layer 4 responder

In the previous section we introduced the Layer 4 responder as one of the interactive deployment types we mount on our address range. Here, we present a description of its implementation and distribution process.

The Layer 4 responder is a powerful tool for obtaining deeper insights into the behavior of internet scanners. Hiesgen et al. introduced Spoki [44], a reactive network component specifically designed to capture the activity of so called *two-phase scanners*, also known as stateless TCP scanners. These scanners not only identify open ports during the first phase of scanning but also initiate a *second scan phase* of further interactions with targeted hosts. While traditional network telescopes effectively capture the first phase of scanning, they fail to observe the second phase due to their passive nature and lack of response to incoming packets. Spoki, instead, aims to respond to incoming TCP SYN packets in real

time, establishing TCP connections, and thereby triggering the second phase of scanning activity. This approach enables the collection of deeper insights into the behaviour of *two-phase scanners*, that otherwise would remain undetected. The study conducted by Hiesgen et al. demonstrated that stateless TCP scanners contribute more than two-thirds of all TCP SYN traffic, and that their activity is significantly more targeted than one-phase scanners.

Implementing a Layer 4 responder requires setting up a system capable of listening for incoming TCP connections on all ports. However, opening all 65,535 ports on a system is both impractical and resource-intensive. To overcome this limitation, we use `iptables` rules to redirect all incoming TCP traffic to a single listening port where the Layer 4 responder operates. To automate this process, we developed a bash script that configures the necessary `iptables` rules. Algorithm 2 provides an overview of the steps performed by the script. The script takes into account: the network interface, the private IP address mapped to the public IP, the listening port of the Layer 4 responder, and the range of ports to be redirected (1-65535). It begins by flushing any existing NAT rules to ensure a clean configuration and enabling IP forwarding to allow traffic redirection. Then, it uses the `multiport` module to add a `DNAT` rule that redirects incoming TCP traffic on the ports in `PORT_LIST` to the `LISTEN_PORT` of the Layer 4 responder.

Algorithm 2 iptables Setup for L4Responder

Require: `NIC`, `PRIVATE_IP`, `LISTEN_PORT`, `PORT_LIST`
`NIC` ▷ Network interface
`PRIVATE_IP` ▷ Private IP address bound to NIC
`LISTEN_PORT` ▷ Port where L4Responder listens
`PORT_LIST` ▷ List or range of ports to redirect

function AddDNATRules(`NIC`, `PRIVATE_IP`, `PORT_LIST`, `LISTEN_PORT`)
    Execute:

```
iptables -t nat -A PREROUTING -d PRIVATE_IP -i NIC -p tcp -m multiport \
   -dports PORT_LIST -j DNAT -to-destination PRIVATE_IP:LISTEN_PORT
```

end function

procedure Setup_iptables
    Flush existing NAT rules
    Enable IP forwarding
    AddDNATRules(`NIC`, `PRIVATE_IP`, `PORT_LIST`, `LISTEN_PORT`)
end procedure

After setting up the `iptables` rules, we developed a Python script to implement the Layer 4 responder functionality. Algorithm 3 provides a high level overview of its main logic. We initiate the responder by starting a TCP server on port `LISTEN_PORT`. However, due to the `iptables` redirection, all incoming TCP traffic appear to have the same destination port, which is the `LISTEN_PORT`, thus losing the original destination port information. To address this, we utilize the `SO_ORIGINAL_DST` socket option to retrieve the original destination IP and port of the incoming

packets. This allows us to know which port the scanner was actually targeting. The responder continuously listens for incoming TCP packets. Upon receiving a packet, it extracts the original destination information and processes the packet based on its type [44]. If the packet is a TCP SYN, the responder sends back a TCP SYN-ACK response, effectively completing the initial handshake. If the packet is a TCP ACK, indicating that the scanner is attempting to establish a connection, the responder sends a TCP RST response to terminate it. If no further packets are received within a predefined timeout period, the responder also sends a TCP RST to close the connection. All other types of packets are ignored.

Algorithm 3 L4Responder with connection timeout handling

```
Require: PRIVATE_IP, LISTEN_PORT, TIMEOUT
PRIVATE_IP                                          ▷ Private IP address
LISTEN_PORT                                         ▷ Port where L4Responder is active
TIMEOUT                                             ▷ Maximum inactivity duration before closing connection

procedure START_L4RESPONDER
    Start TCP server on PRIVATE_IP:LISTEN_PORT
    while True do
        Wait for incoming TCP connection
        original_dst ← Extract original destination using SO_ORIGINAL_DST
        Handle connection with REPLY_RULES(connection, original_dst)
    end while
end procedure

procedure REPLY_RULES(connection, original_dst)
    if received TCP SYN then
        Send TCP SYN-ACK response
    else if no packets received for TIMEOUT seconds then
        Send TCP RST and close connection
    else if received TCP ACK then
        Send TCP RST response
    else
        Ignore packet
    end if
end procedure
```

To streamline the deployment process, we containerized the Layer 4 responder application using Docker. The Docker image includes the Python script, all necessary dependencies, and the bash script for configuring `iptables` rules. The Docker container runs in privileged mode with `host network` access, allowing it to directly modify `iptables` rules on the host system.

Chapter 5

Data Collection and Analysis

In this chapter, we analyze the traffic volumes and characteristics observed in a cloud-based network telescope and compare them with those of a traditional telescope hosted on our campus IP space. We begin by presenting the data collection campaign, detailing different time windows with specific configurations, including the deployment of subnets hosting fake services with varying levels of interactivity, such as Layer 4 responders and honeypots. Next, we perform a comprehensive statistical analysis of the collected data, highlighting the differences between the two telescopes. Our analysis focuses on traffic volumes, timeseries of incoming traffic, impact of interactive services, and the behavior of senders targeting each deployment.

5.1 Data collection campaign

For this study, we collected one month of traffic from two network telescopes: one hosted in the Azure cloud environment and the other on our campus IP space. In Section (4.3.2), we discussed the adoption of various interactive deployments designed to enhance the visibility of the telescopes and provided details about their subnet allocation. In this section, we outline the timeline of our data collection campaign, highlighting the distinct time windows and specific deployment configurations used during each phase.

5.1.1 Timeline data collection

In this section, we outline the timeline of our data collection campaign, detailing the overall period and the specific subperiods during which different interactive deployments operated. Figure 5.1 illustrates the timeline, which we divided into four distinct phases.

- **Phase 1: (July 25 - August 5).** During this initial phase, all deployments were inactive, resulting in a full darknet configuration on both the campus and cloud telescopes. This phase serves as a baseline for observing unsolicited traffic in the absence of interactive services.
- **Phase 2: (August 5 - August 15).** In this phase, we activated the predefined Layer 4 responder and honeypot deployments, enabling them to respond to incoming traffic. Additionally, we configured specific cloud subnets with DNS announcements and public certificate emissions, for enhance their visibility. This phase allows us to evaluate the impact of interactive deployments on observed traffic, as well as the effect of DNS announcements and public certificate emissions with respect to similar deployments.
- **Phase 3: (August 15 - August 15).** In this phase, we deactivated all interactive deployments, reverting to a full darknet configuration. However, DNS announcements and public certificate emissions remained enabled throughout the entire campaign to evaluate their residual impact. Also, we kept ICMP responses active only on cloud telescope to assess how this influences incoming traffic. This phase allows us to evaluate whether active deployments had any lasting effect on the traffic observed when both telescopes reverted to a darknet configuration.
- **Phase 4: (August 25 - September 2).** Following these three phases, we continued data collection until budget expiration on September 2. During this extended period, we reactivated the services on to observe whether any significant changes occurred compared to the initial activation phase.

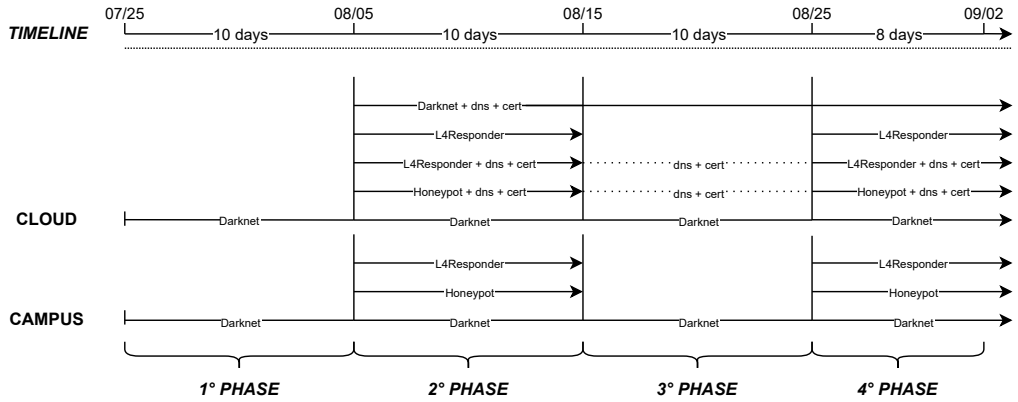


Figure 5.1: Data collection timeline

This structured timeline enables a comparative analysis of the cloud and campus

telescopes under different configurations. In the following sections, we focus on the first two phases to analyze traffic volumes, examining both passive and interactive deployments and the impact of interactive services. Additionally, we evaluate the subsequent phases to assess the lasting effects of interactive services and the emergence of new senders.

5.2 Statistical analysis

In this section, we present a statistical analysis of the collected data to identify differences between the cloud-based telescope and the traditional campus telescope. We begin by analyzing the traffic volumes observed in both passive and interactive deployments. Then, we examine the impact of activating responders on both telescopes by visualizing the time series of traffic volumes and focusing on the amplification effects introduced by these responders. We then analyze the lasting impact of interactive deployments after their deactivation, assessing whether their influence persists over time. Finally, we conduct a sender analysis for both the cloud and campus deployments, exploring the distribution of source IPs, identifying unique and common sources, and evaluating whether interactive deployments attract new sources of traffic or amplify the activity of already present ones.

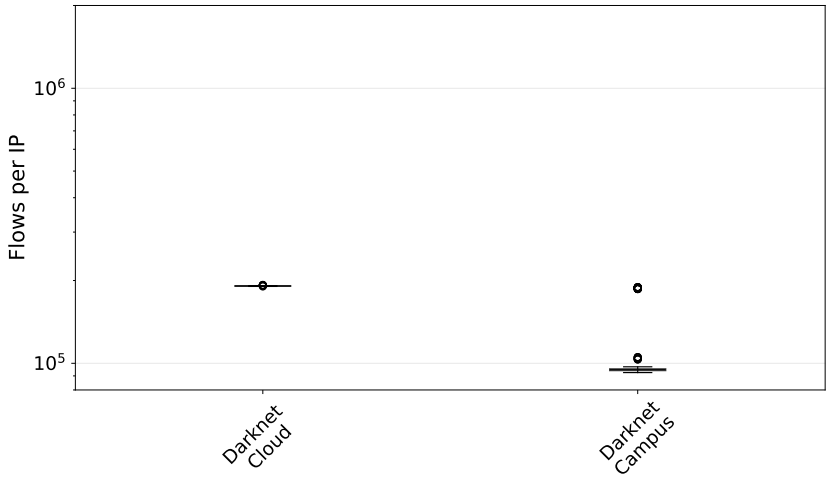
Due to the interactive deployments, the collected traffic consists of both incoming and outgoing packets. To facilitate analysis, we segment the traffic into flows. A flow is defined as a bidirectional 5-tuple consisting of the source and destination IP addresses, source and destination ports, and the transport-layer protocol. Packets sharing the same 5-tuple are grouped into the same flow, provided the idle time between packets does not exceed 5 minutes. If two packets with the same 5-tuple arrive more than 5 minutes apart, they are treated as separate flows.

5.2.1 Traffic volume

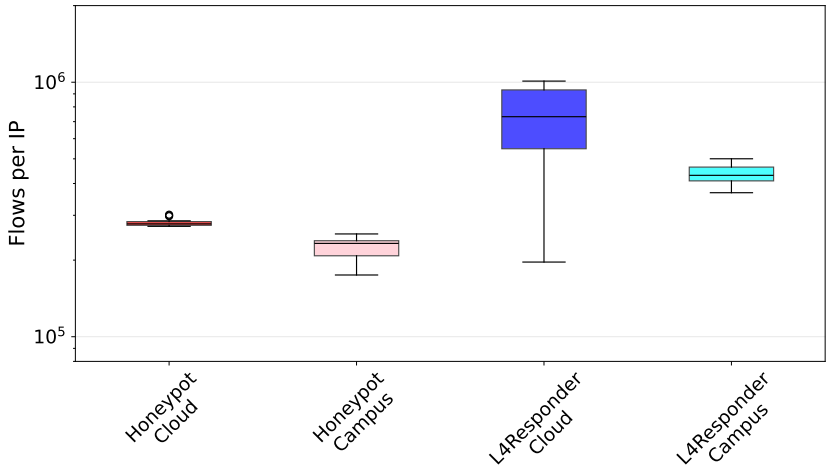
In this section, we analyze the differences in traffic volumes collected by the two telescopes, considering both passive and active deployments. Specifically, for passive deployments, we compare the traffic volumes observed on darknet subnets in the first phase time window (July 25 - August 5th), where all deployments were inactive. For active deployments, we compare the traffic volumes observed on interactive deployments (Layer 4 responder and Honeypot) in the second phase time window (August 5 - August 15th), when these services were active. Figure 5.2 shows the results. The x-axis represents the specific subnet deployment, while the y-axis shows the average number of flows received by each address in the subnet.

Figure 5.2a reveals that the cloud darknet receives nearly twice the number of flows per IP compared to the campus darknet. However, the campus darknet exhibits some outliers with significantly high traffic volumes. This anomaly is

likely due to the campus IP space previously hosting active services, which may have attracted more targeted traffic. Figure 5.2b shows a similar trend with cloud deployments receiving more flows than campus counterpart. While the difference is more marginal for Honeypots deployments, it is more pronounced for Layer 4 responder deployments.



(a) Darknet volumes per IP



(b) Interactive Deployments volumes per IP

Figure 5.2: Traffic volumes comparison

Thus, the results revealed that the cloud telescope consistently attract more traffic than the campus telescope, regardless of the deployment type. Now, we aim to analyze the effects on traffic volume brought by the activation of interactive services on both telescopes.

5.2.2 Timeseries

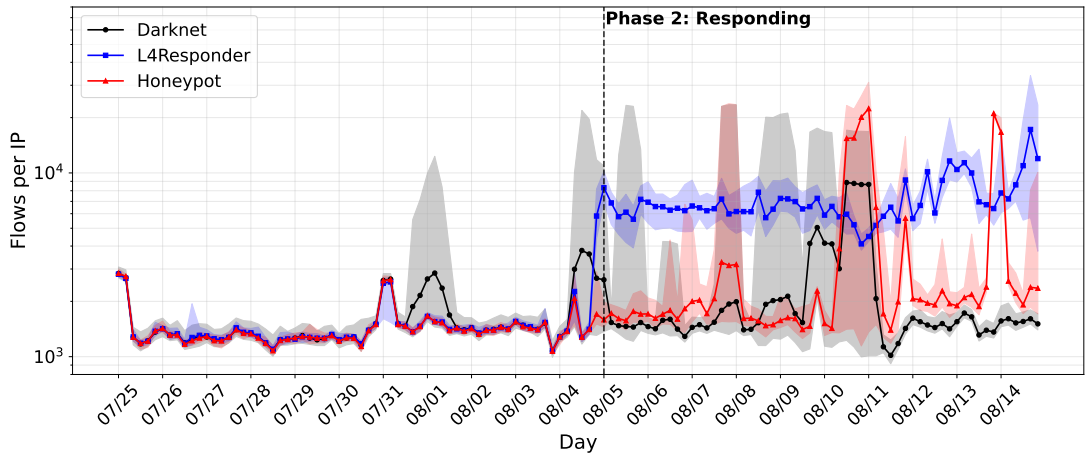
In Section (5.1.1), we outlined the different phases of our data collection campaign, each with specific deployments configuration. In this section, we evaluate the impact of interactive deployments on the overall traffic volumes observed by both telescopes, focusing on the traffic received during the first and second phase time windows. Figure 5.3 illustrates the time series of traffic volumes observed across the different subnet deployments within the campus and cloud IP space. The x-axis represents the day of the month, while the y-axis shows the average number of flows received by each address in the subnet deployment, aggregated over 4-hour intervals. The shaded areas indicate the minimum and maximum values observed within each interval, while the vertical dashed line marks the transition between the first phase (darknet configuration) and the second phase (activation of Layer 4 responder and honeypot).

During the first phase, when all subnets operate in a darknet configuration, the number of received flows is relatively similar across deployments on both the campus (Figure 5.3a) and cloud (Figure 5.3b) networks. On the campus network, the average traffic volume is approximately 1.5×10^3 flows per 4-hour interval, whereas on the cloud network, it is nearly double, reaching around 3×10^3 flows per time bin. Notable peaks occurred on July 31st and August 1st on the campus network (Figure 5.3a). While the peak on July 31st affected all subnets, the peak on August 1st primarily targeted the darknet deployment. This likely reflects the outliers observed in Figure 5.2a.

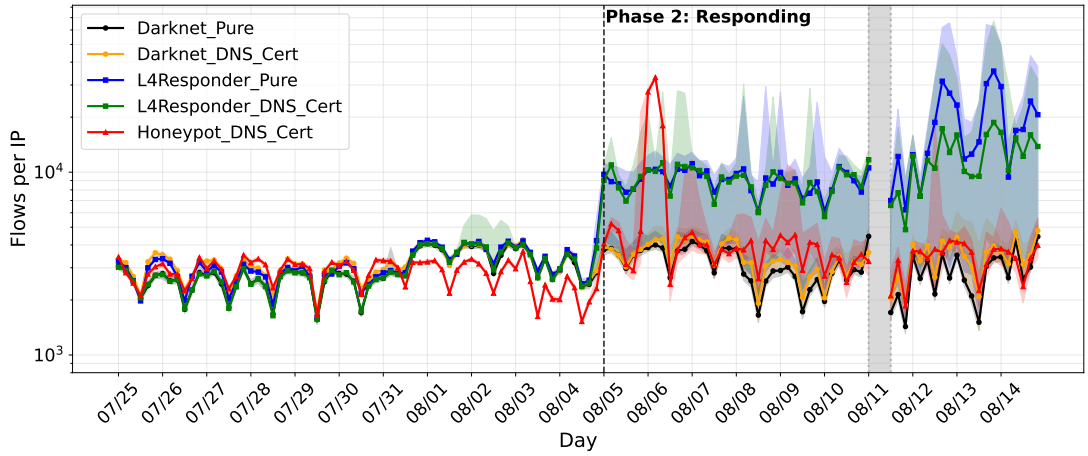
On August 5th, we activated the responders on both networks. While the darknet deployments maintained traffic levels similar to those observed in the first phase, with the exception of occasional peaks, the impact of this change became immediately evident in the Layer 4 responders. Notably, signs of increased activity on the campus network appeared on August 4th, as the activation process started on this day and reached full activation by August 5th. While in the first phase, the Layer 4 responder exhibited traffic levels comparable to other deployments, in the second phase, it consistently receives significantly more incoming flows, increasing to 7×10^3 on the campus and 9×10^3 on the cloud. Interestingly, on both networks, we observe a further increase in the number of flows targeting the Layer 4 responder starting from August 11-12th. This trend suggests that the Layer 4 responder continued to attract increasing attention from scanners, further widening the gap between its traffic volume and that of other deployments over time.

In contrast, the honeypot deployment shows a more modest increase in traffic compared to the Layer 4 responder. This behavior is expected, as the Honeypot service is only active on ports 22, 80, and 443, whereas the Layer 4 responder operates on all ports (1-65535). During the first phase, the traffic volumes for the Honeypot and Darknet deployments are nearly identical. However, in the second

phase, the Honeypot consistently receives more traffic than the Darknet, indicating a slight increase in visibility.



(a) Campus



(b) Cloud

Figure 5.3: Traffic volume over time for deployments (flows/4 hours)

By the August 5th, we also enabled DNS announcements and public certificate emissions for specific subnets in the cloud telescope. Figure 5.3b allows us to assess, between similar deployments, how this factor influences the visibility of the subnet. The two Layer 4 responders show a similar trend until the August 11-12th, when the L4Responder_Pure starts to receive more traffic than the L4Responder_DNS_Cert. This behavior would suggest that the DNS announcements and public certificate emissions do not bring significant advantage for receiving more traffic. However, it is important to note that the two Layer 4 responders reside in different IP

ranges (see Table 4.3a), as we were unable to achieve full /24 subnet contiguity due to cloud provider constraints. Therefore, this effect might be the result of the different intrinsic nature of the two range locations. On the other hand, the two darknets (Darknet_Pure and Darknet_DNS_Cert), exhibit a different behavior. The Darknet_DNS_Cert consistently receives slightly more traffic than the Darknet_Pure. However, we can note this difference already during the first phase, when none of the two darknets had DNS announcements or public certificate emissions active. For this reason, as the previous case, we cannot attribute any significant benefit to the presence of DNS announcements and public certificate emissions.

On the cloud network, during the first phase time window, we observe noticeable drops in traffic occurring in the middle of each day. Figure 5.4 provides a clearer view of this phenomenon. It shows the activity of the top 500 source IPs generating traffic for a sample /28 subnet of the cloud darknet deployment during the first phase time window. These top 500 IPs account for more than 50% of the traffic within that subnet. Each row represents the activity of a source IP, segmented into 4-hour time bins. The graph reveals that these traffic drops are caused by a significant group of senders that consistently cease their activity during the same hours in the middle of each day, exhibiting a distinct and synchronized pattern.

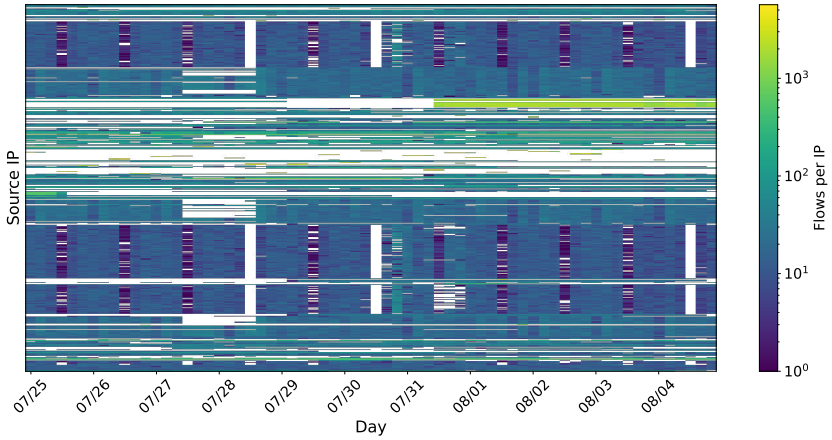
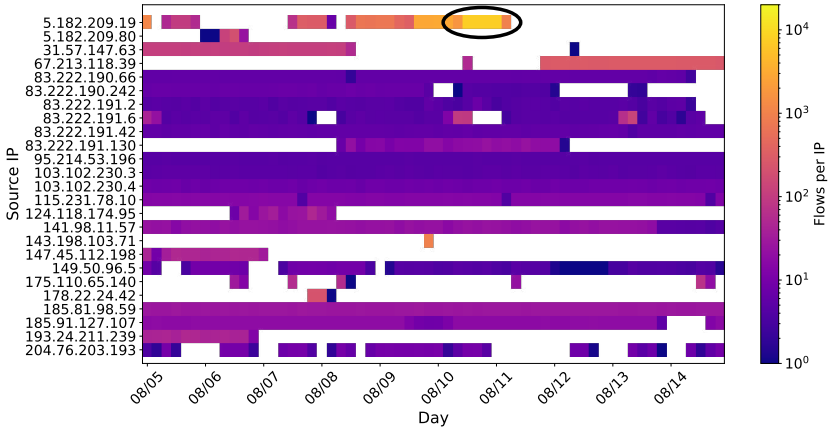


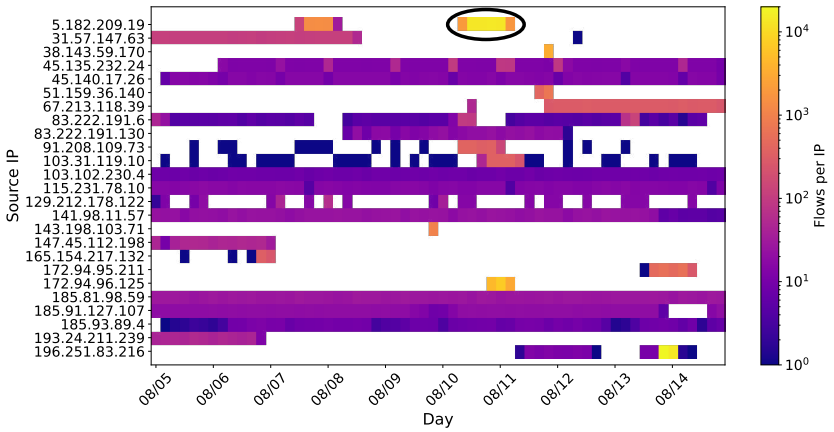
Figure 5.4: Activity of the top 500 source IPs for a sample /28 Cloud Darknet deployment (flows per 4-hour interval)

On campus network, we observe visible peaks involving honeypot and darknet deployments on August 10-11th. Figure 5.5 highlights the top 25 source IPs contributing to the traffic for darknet and honeypot deployment during the same period. Each row represents the activity of a source IP, segmented into 4-hour time bins. The color intensity indicates the average number of flows sent by a source IP to each address in the subnet deployment within the time bin.

Specifically, we observe a high intensity of flows originating from the source IP address "5.182.209.19" on August 10-11th. This sender is likely responsible for both the peaks observed for darknet and honeypot subnets. This observation underscores an important consideration: due to the contiguity of the campus subnet, some scanners may have scanned the adjacent Darknet deployment after identifying open ports in the Honeypot deployment. As a result, even though the Darknet deployment is inactive, it may experience a *side effect* caused by the proximity of the active services. Figure 5.5b also depicts the activity of the IP "196.251.83.216", responsible for a peak observed on August 13-14th in the honeypot subnet.



(a) Campus Darknet



(b) Campus Honeypot

Figure 5.5: Top 25 source IPs activity for Campus Darknet, Campus Honeypot and Cloud Honeypot deployments (flows/4 hours)

Lasting effects

In the previous section, we analyzed the impact brought by the activation of interactive services on the traffic volumes observed by both telescopes. Now, we aim to evaluate whether these effects persist even after deactivating such services. In this section, we analyze the period following the first two phases of the data collection campaign. In Section (5.1.1), we defined that the third phase of the campaign involved turning off all interactive deployments. However, we kept certain factors active for the cloud deployments, including DNS announcements, public certificate emissions, and ICMP responses.

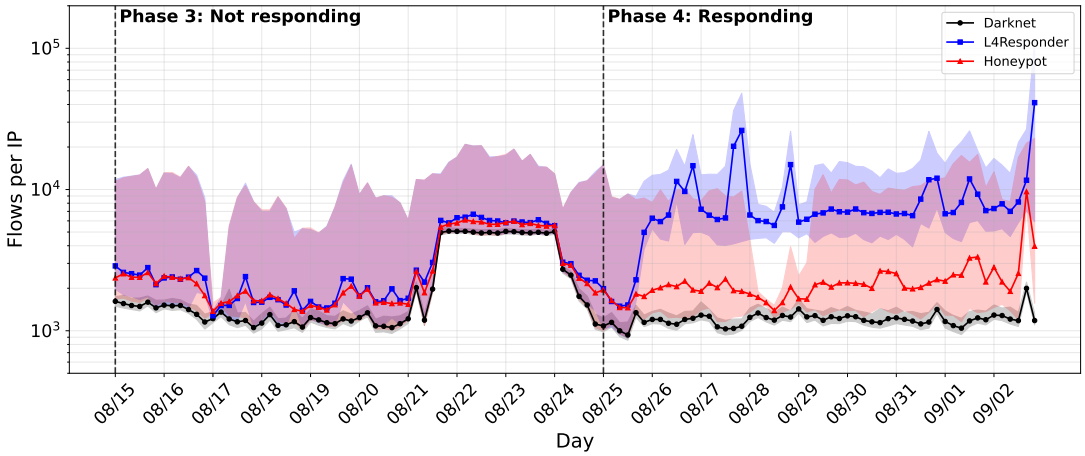
In Figure 5.3, we observed the time series of traffic volumes observed across the different subnet deployments within the campus and cloud IP space, respectively, across the first and second phase time windows. Figure 5.6 extends this analysis to include the subsequent period until September 2nd. This consists of a third phase (August 15-25th), and an additional fourth phase (August 25 - September 2nd) where we reactivated the interactive deployments.

Figure 5.6a shows that, after deactivating the interactive deployments, the traffic volumes on all campus subnets return to levels comparable to those observed during the first phase. Notably, while in the first phase the Layer 4 responder, honeypot, and darknet exhibited nearly overlapping traffic volumes (see Figure 5.3a), the third phase reveals a marked difference. Specifically, the Layer 4 responder receives the highest volume of traffic, followed by the honeypot and then the darknet. This observation suggests that, despite the deactivation of interactive deployments, scanners may have retained knowledge of the previously active services, leading them to continue targeting the subnets that hosted these services. We, interestingly note that during the time interval from August 21st to August 24th, a significant increase in received flows across all campus subnet deployments occurs.

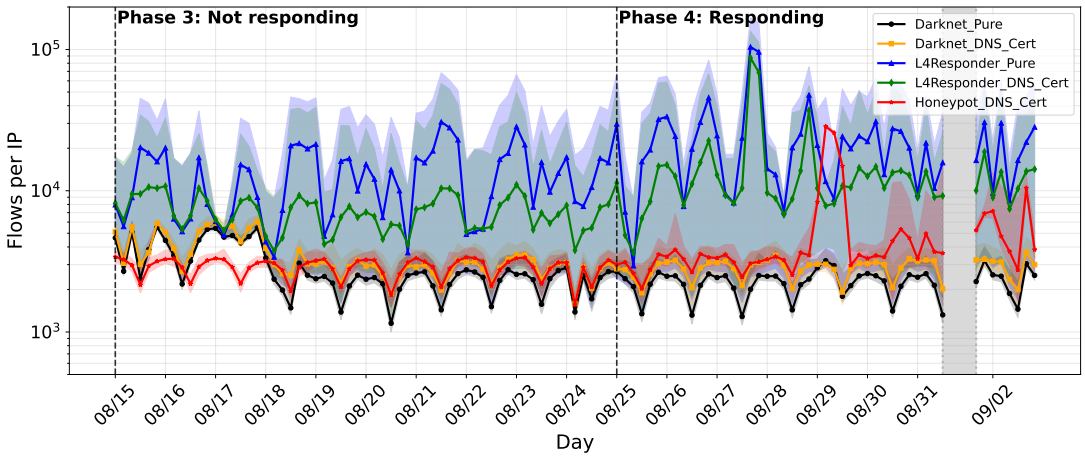
On the cloud side, the number of flows reaching each deployment remains consistent despite the deactivation of interactive services (Figure 5.6b). It is important to note that the cloud deployments continued to have DNS announcements, public certificate emissions, and ICMP responses active during this period. While we can reasonably exclude the influence of DNS announcements and public certificate emissions, as these factors did not demonstrate a significant impact during the second phase, it is plausible that the ICMP responses played a role in maintaining the visibility of the cloud deployments. These represent signals that the subnet are actually up, thus leading scanners to continue targeting them.

On the campus telescope, as we observed during the second phase, reactivating the interactive deployments on August 25th resulted in an immediate surge in traffic with levels comparable to those observed during the initial activation phase. Additionally, we observe the gap between the honeypot and darknet deployments to be widening progressively over time. In contrast, on the cloud telescope, reactivating

the interactive deployments did not result in any significant change in traffic volumes with respect to the third phase time window. This is normal as the traffic volumes remained consistent even during the third phase.



(a) Campus



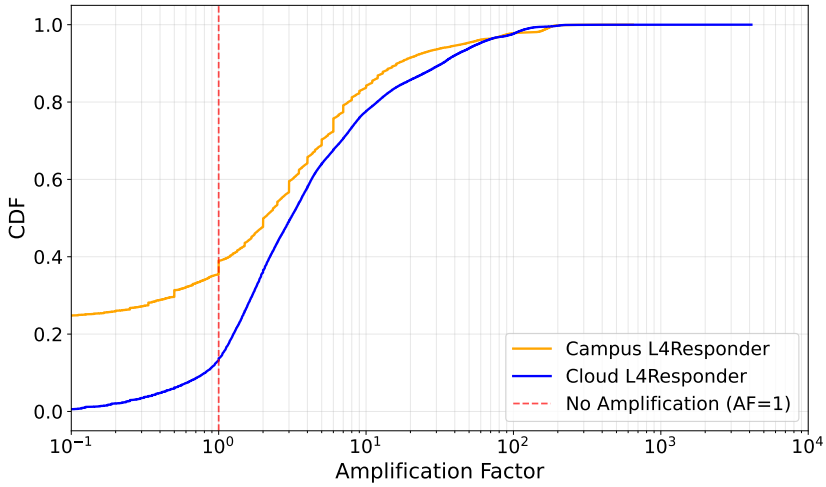
(b) Cloud

Figure 5.6: Time series of traffic volumes during 3° and 4° phase (flows/4 hours)

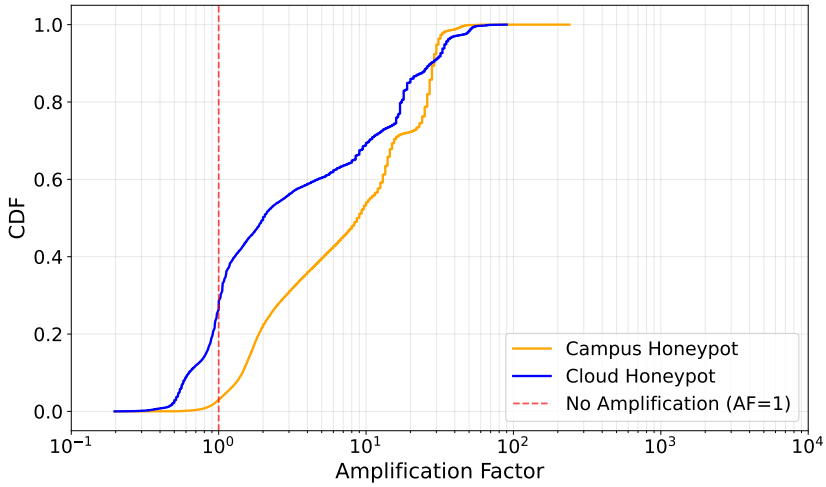
5.2.3 Port amplification factor

In the previous section, we observed that activating services leads to receive more traffic. Now, we want to quantify the amplification effect brought by the activation of such services. The amplification factor is defined as the ratio between the number of flows targeting a specific port during the second phase (when interactive

deployments are active) and the number of flows targeting the same port during the first phase (when the same deployments are in darknet configuration). This metric allows us to quantify how much more traffic a specific port receives when it is associated with an active service compared to when it is inactive. Figure 5.7 shows the cumulative distribution function (CDF) of port amplification factors for both Layer 4 responder (Figure 5.7a) and Honeypot deployments (Figure 5.7b) on campus and cloud IP spaces. The x-axis represents the port amplification factor, while the y-axis indicates the cumulative fraction of destination ports.



(a) AF for Layer 4 responder deployements



(b) AF for Honeypot deployments

Figure 5.7: Cumulative Distribution Function (CDF) of Amplification Factors by Destination Ports for Layer 4 Responder and Honeypot Deployments

Figure 5.7a shows that the curve for the cloud Layer 4 responder consistently lies to the right of its campus counterpart, indicating a stronger amplification effect. Specifically, 85% of the destination ports on the cloud responder experience an amplification, compared to 60% on the campus responder. In contrast, Figure 5.7b reveals a different trend. While the cloud honeypot exhibits a higher peak amplification in the upper 5 percentiles, the campus honeypot demonstrates a generally stronger amplification effect overall. Approximately 97% of the destination ports on the campus honeypot experience amplification, whereas this percentage drops to around 76% for the cloud honeypot.

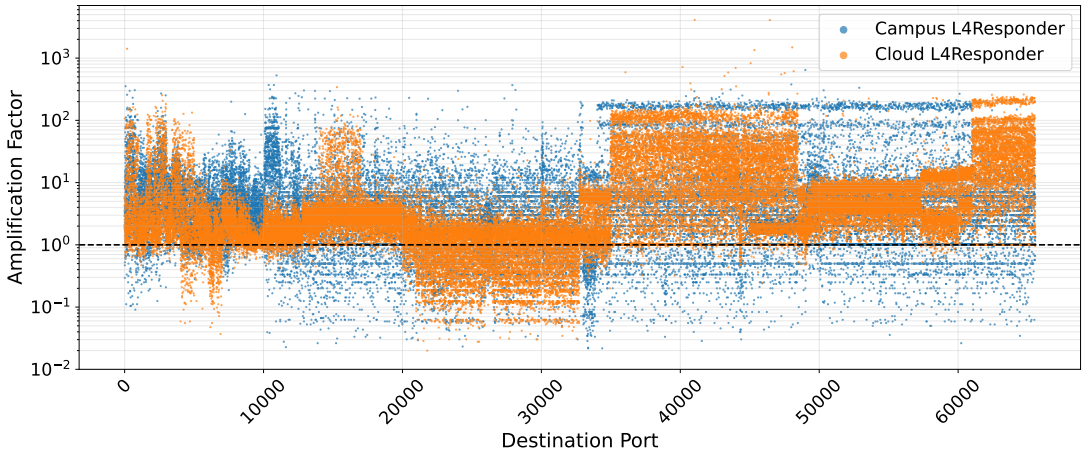
These results indicate that the amplification effect introduced by activating interactive services varies depending on the type of deployment. For the Layer 4 responder, the cloud shows a stronger overall amplification effect compared to the campus counterpart. On the contrary, for the honeypot deployment, the campus demonstrates a more pronounced amplification effect across the wide range of ports.

To further analyze the port amplification factor between cloud and campus deployments, we compute the amplification factor for each destination port and visualize the results using scatter plots for both the Layer 4 responder (Figure 5.8a) and Honeypot (Figure 5.8b). The x-axis represents the destination port number, while the y-axis indicates the observed amplification factor for that port. This visualization allows us to identify what is the general amplification effect on different port ranges, and to compare the behavior between cloud and campus deployments.

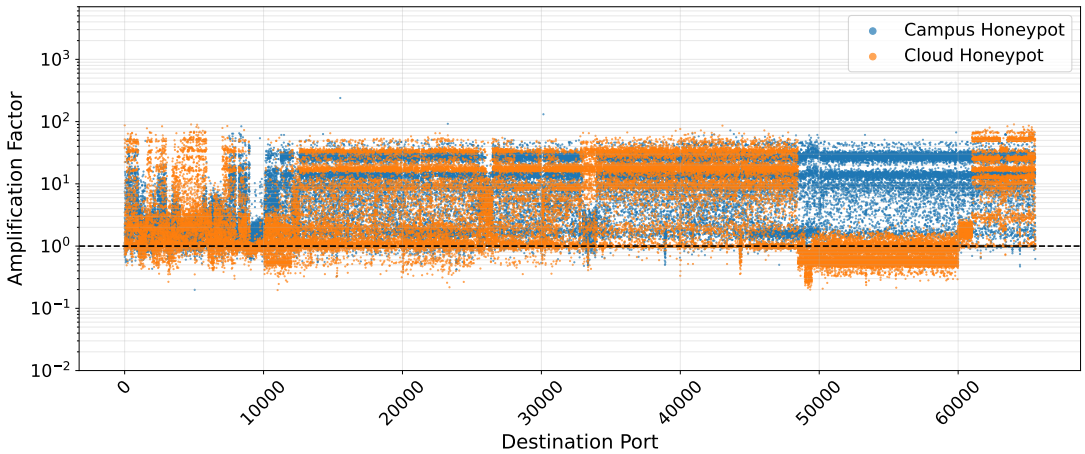
The cloud Layer 4 responder demonstrates a relatively homogeneous amplification factor across the entire range of ports, where similar ports exhibit similar amplification. In contrast, the campus Layer 4 responder exhibits a more heterogeneous behavior, where certain port ranges contain ports with amplification factors both above and below one. The cloud Layer 4 responder generally shows positive amplification factors for most port ranges, with highest amplification observed in the ranges [0-20,000], [33,000-48,000], and [60,000-65,535]. Notably, it exhibits amplification factors below one in the ranges [20,000-33,000], thus indicating a reduction in traffic attraction for these ports. While this might seem to be a counterintuitive result, this is not unusual. We observed something similar in [45] where the authors found the senders of Layer 4 responder to skip ports around port 30,000. This behavior is likely a consequence of enabling all ports, which causes scanners to focus on certain ports while neglecting others, possibly biasing their activity.

For honeypot deployments (Figure 5.8b), instead, the cloud and campus exhibit a generally similar trend across most port ranges. Both show amplification factors around the order of 3×10^3 for most of the ports. However, a difference emerge in the [50,000-60,000] port range, where the campus honeypot demonstrates a stronger amplification effect, while the cloud honeypot shows amplification factors

below one. Conversely, for the first 8,000 ports, the cloud honeypot exhibits a more pronounced amplification effect compared to the campus honeypot.



(a) Amplification Factor for L4 Responder



(b) Amplification Factor for Honeypot

Figure 5.8: Port amplification factor for L4Responder and Honeypot deployments

5.2.4 Sender analysis

In this section, we focus on understanding which senders are responsible for this unsolicited traffic. We analyze the distribution of source IPs contacting the two telescopes, the presence of unique and common senders, and whether interactive deployments attract new sources of traffic or amplify the activity of already existing ones.

Source IP distribution

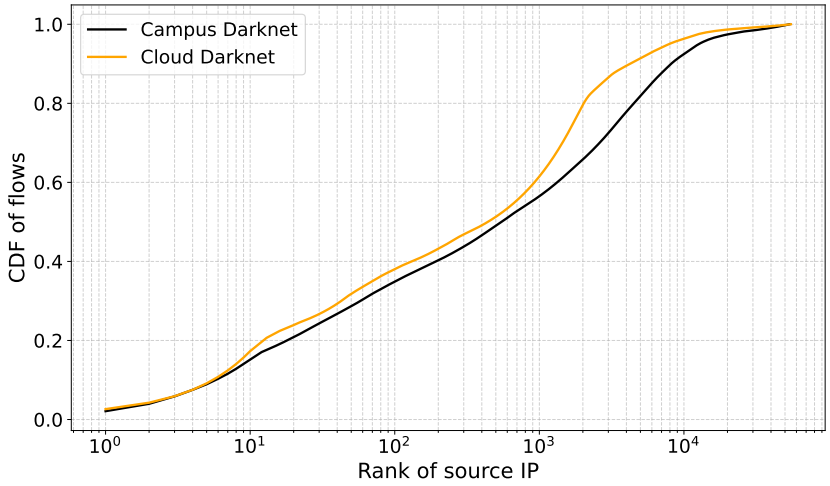
In this section, we compare the flows distribution of source IPs contacting the two telescopes, analyzing both passive and interactive deployments. For the passive deployments, we focus on senders targeting darknet subnets in the first time window, while for the interactive ones, we consider senders targeting interactive subnets on the second time window. Figure 5.9 shows the cumulative fraction of flows by source IP for both the cloud and campus deployments. Table 5.1 provides the number of unique source IPs observed in each deployment, along with the breakdown of senders that contact both the campus and cloud telescopes, referred to as "common" senders, and those that target only one of them, referred to as "exclusive" senders.

Phase	Deployment Type	Unique Source IPs	Common IPs	Exclusive IPs
First Phase	Campus Darknet (sample /28)	53,701	24,724	28,977
	Cloud Darknet (sample /28)	54,912		30,188
Second Phase	Campus L4Responder	69,167	32,621	36,546
	Cloud L4Responder	78,056		45,435
	Campus Honeypot	54,177	28,204	25,973
	Cloud Honeypot	63,208		35,004

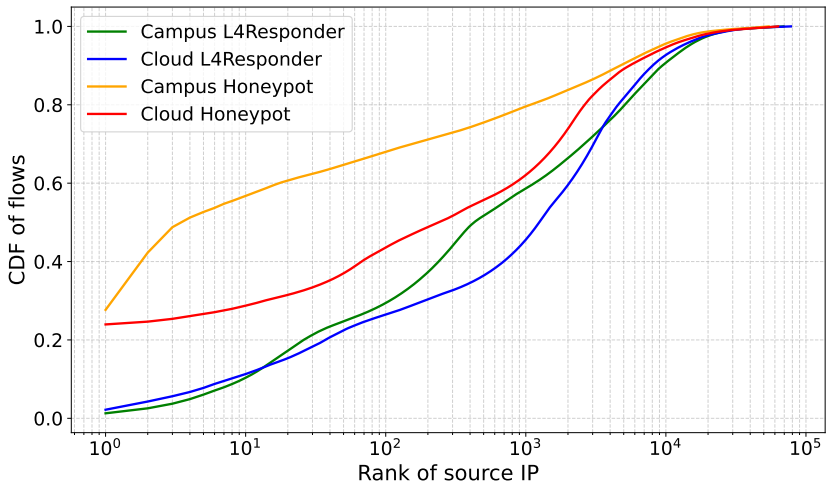
Table 5.1: Unique Source IPs per Deployment and Phase

It is important to note that, due to the subdivision of deployments (Table 4.3a, 4.3b), the cloud darknet encompasses 192 IP addresses, while the campus darknet includes 224. To ensure a fair comparison, we normalize the number of IP addresses by sampling a /28 subnet from both darknet deployments. The two deployments show a difference of 1,211 unique source IPs, with the cloud darknet observing 54,912 unique source IPs and the campus darknet observing 53,701. This result is noteworthy, especially when compared to the findings of [3], where the authors reported that the cloud darknet received traffic from 73% more IP addresses than the traditional darknet. The CDF reveals a similar distribution among the top-ranked source IPs targeting campus and cloud darknets, with the top 20 IPs contributing roughly 20% of the total traffic in both cases. However, the distributions begin to diverge significantly after the top 1,000 IPs, where cloud darknet is more influenced by a group of highly active source IPs.

While the two darknet deployments receive traffic from a similar number of unique source IPs, with interactive deployments a difference emerge with cloud telescope receiving traffic from more then 10,000 additional unique source IPs (see Table 5.1). Figure 5.9b reveals that Campus deployments are more influenced by a small group of highly active source IPs. This is particularly evident for the Honeypot deployment, where top 3 IPs contribute to the 50% of the total traffic.



(a) Darknet flows CDF



(b) Interactive deployments flows CDF

Figure 5.9: Cumulative fraction of flows by source IP (Campus vs Cloud)

Cloud honeypot requires 200 IPs to reach the same percentage. On the other hand, the two Layer-4 responders show a more similar distribution, with the campus deployment being slightly more influenced by highly active IPs. However, the difference is not as pronounced as in the Honeypot case.

Common vs exclusive senders

In this section, we evaluate the contributions of "common" and "exclusive" sender categories to overall traffic and compare how this distribution differs between

cloud and campus deployments. Table 5.1 shows that, for most deployments, the majority of unique source IPs are exclusive to each telescope, with the cloud telescope consistently observing a higher number of exclusive senders. Table 5.2 presents the fraction of flows generated by each sender category. For darknet and Layer 4 responder deployments, most traffic originates from common senders, whereas for honeypot deployments, exclusive senders contribute more significantly. Therefore, although most unique senders across deployments are exclusive to each telescope, common senders have a larger share of overall traffic. Notably, across all deployments, common senders consistently generate more flows to the cloud telescope compared to the campus counterpart.

Deployment	**Total Flows**	**By common sender**	**By exclusive sender**
Campus Darknet	1,521,477	1,104,554 (72.6%)	416,923 (27.4%)
Cloud Darknet	3,055,385	1,880,230 (61.5%)	1,175,155 (38.5%)
Campus L4Responder	6,920,754	5,078,767 (73.4%)	1,841,987 (26.6%)
Cloud L4Responder	11,596,348	7,951,212 (68.6%)	3,645,136 (31.4%)
Campus Honeypot	3,593,698	1,242,780 (34.6%)	2,350,918 (65.4%)
Cloud Honeypot	4,494,583	1,988,958 (44.3%)	2,505,625 (55.7%)

Table 5.2: Traffic Contribution by Common and Exclusive Senders

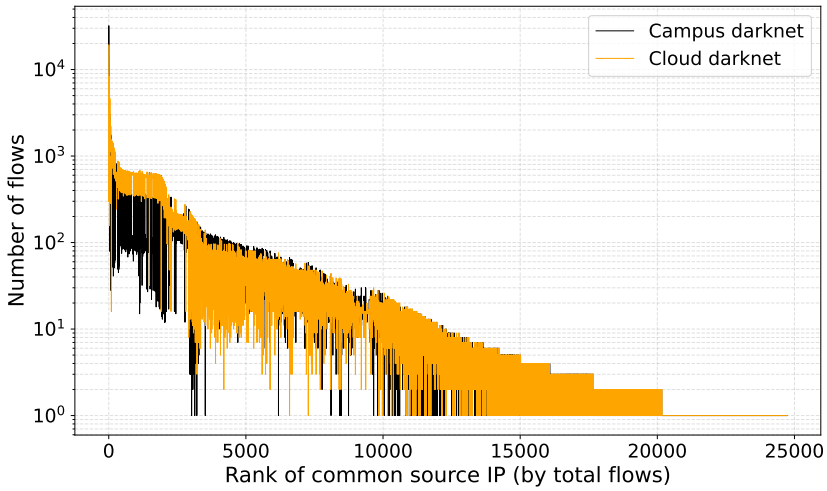


Figure 5.10: Darknet common sender flows comparison

Figure 5.10 further investigates the behavior of common senders. It shows the number of flows sent by common source IPs to the campus and cloud darknet deployments, ranked by number of total flows. We observe that among the common senders with the highest activity, large part of them target the cloud telescopes

with more traffic than the campus counterpart. However, as we move down the ranking, we observe a more balanced distribution, with some senders being more active on the campus darknet.

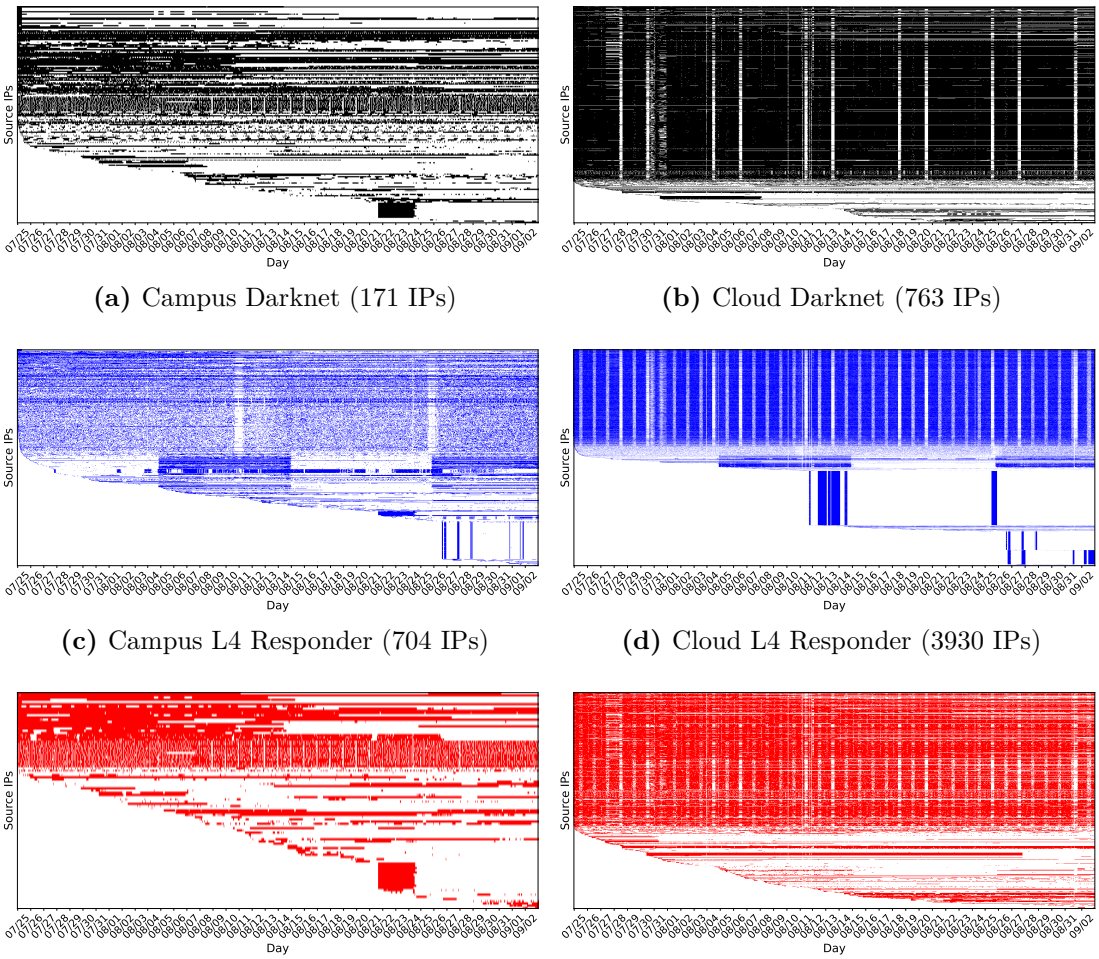
New senders arrival

In this section, we analyze whether the interactive services attract new source IPs or amplify the activity of existing ones. Figure 5.11 visualizes the activity of the most active senders contributing to 50% of the total TCP flows for each specific deployment on the campus and cloud networks over the entire observation period (July 25th - September 2nd). Each row in the graph represents the activity of a single sender, with a point marking the specific hour when the sender was active. The graph sorts the senders by their time of first appearance, enabling us to observe the emergence of new groups of senders and track their activity over time.

Figures 5.11a and 5.11b illustrate the activity of source IPs targeting the campus and cloud darknets, respectively. On the cloud darknet, many senders remain active throughout the observation period, with notable surges of new senders appearing on July 31st and August 15th. In contrast, the campus darknet shows a more sporadic activity pattern, with senders appearing intermittently over time. A significant group of senders becomes active between August 21st and August 24th, coinciding with the traffic spike observed in Figure 5.6a. We notice a similar event in the other two campus deployments (Figures 5.11c and 5.11e), suggesting that this surge in activity was a widespread event affecting the entire campus network.

The Layer 4 responder deployments highlight the impact of enabling interactive services. Both campus and cloud deployments experience significant increases in the number of active senders during the two activation phases (August 5-15th and August 25-September 2nd). These increases result both from the arrival of new senders and from intensified activity from existing senders. Notably, on both deployments, the same new senders that appeared during the first activation phase reemerge during the second activation phase, while reducing their activity during the deactivation period (August 15-25th). On the cloud Layer 4 responder (Figure 5.11d), a large influx of new senders occurs on August 11th-12th, coinciding with the traffic increase observed in Figure 5.3b, where the L4Responder_Pure begins to receive more traffic than the L4Responder_DNS_Cert. Additionally, both campus and cloud Layer 4 responder deployments observe the arrival of new senders around August 26th, exhibiting similar time patterns. This similarity suggests that these senders may be targeting both telescopes.

Finally, the honeypot deployments (Figures 5.11e and 5.11f) show a gradual arrival of new senders over time. On the campus honeypot, we observe a group of senders becoming active on August 5th and remaining consistently active until the end of the observation period. We observe similar things on the cloud counterpart.



(a) Campus Darknet (171 IPs)

(b) Cloud Darknet (763 IPs)

(c) Campus L4 Responder (704 IPs)

(d) Cloud L4 Responder (3930 IPs)

(e) Campus Honeypot (98 IPs)

(f) Cloud Honeypot Sender Activity (620 IPs)

Figure 5.11: Senders activity on Campus and Cloud deployments

5.3 Summary of results

In this section, we summarize the key findings from our comparative analysis of campus and cloud telescopes, focusing on traffic volumes, the effects of different interactive deployments, and sender characteristics.

Our analysis revealed that a cloud-based network telescope receive barely double the amount of flows compared to a traditional campus-based telescope. We, then investigated the effects of enabling interactive deployments. After having

observed the timeline of traffic flows targeting each deployment, we evaluated the amplification effect. We determined that the cloud telescope exhibit an higher amplification effect on the Layer 4 responder, while the campus telescope shows a stronger amplification effect on the honeypot deployment. We also noticed that deactivating interactive services does not completely eliminate their effects, as on campus telescope, while exhibiting a generally lower traffic, the Layer 4 responder continues to receive more traffic followed by the honeypot and then the darknet deployment. On the cloud telescope, instead, we kept certain factors active, such as DNS announcements, public certificate emissions, and ICMP responses. While the former did not represent a significant factor, the ICMP responses likely contributed to maintaining the visibility of the cloud deployments, maintaining the level of traffic consistent with the previous phase.

Finally, we evaluated the diversity of sources contacting the two telescopes. Both the campus and cloud deployments receive traffic from a similar number of unique source IPs and share a significant portion of common senders. However, the cloud telescope consistently receives more traffic from these common senders and attracts a larger number of exclusive senders i.e., sources that contact only the cloud telescope and not the campus one.

Chapter 6

Frequent pattern mining for traffic analysis

In this chapter, we examine the use of data mining techniques to uncover patterns within network traffic data. We detail our approach for effectively applying frequent pattern mining algorithms. First, we provide an overall characterization of internet scanning activities targeting specific subnet deployments, followed by an analysis of the behavior of well-known host groups.

6.1 Setup and basis for the analysis

In this section, we outline the approach used to prepare the data for frequent pattern mining. We start by introducing the libraries and algorithms considered for the analysis. Next, we discuss the fundamental elements of frequent pattern mining, including transaction types and mining techniques. Then, we evaluate these two aspects to determine the most suitable approach for our analysis. Finally, we describe the steps involved in data preprocessing, transaction construction, and the execution of the selected algorithm.

6.1.1 Algorithms and libraries

Network telescopes collect vast amounts of data, making it essential to use scalable and efficient algorithms to extract results that are both meaningful and computationally feasible. In this section, we outline the libraries and algorithms employed in our analysis.

We used *FP-Growth* [46], a well-known algorithm for mining frequent itemsets and association rules. This is widely regarded as the state-of-the-art baseline in frequent pattern mining, and is noted for its efficiency and scalability. We

implemented the algorithm using the *Pyfim* library [47], which provides an efficient implementation of *FP-Growth* and is capable of handling large datasets effectively.

6.1.2 Techniques approach

To apply frequent pattern mining algorithms, it is necessary to transform the network traffic data into a suitable format. These algorithms require the data to be organized into transactions, where each transaction is represented as a set of items. The composition of such transactions defines the scope of the analysis, as it determines the set of items that the algorithm will evaluate to extract frequent patterns. In this study, we explored two different approaches for composing transactions. Table 6.1 provides examples of both.

- **Row transactions**. In this approach, each transaction corresponds to a single row in the dataset, representing an individual network packet. The items in the transaction include attributes such as the source IP, destination IP, source port, destination port, protocol, and other relevant features. This method enables the extraction of patterns related to how senders construct individual packets, such as the protocols they use, typical packet lengths, and commonly targeted ports.
- **Value transactions**. In this approach, each transaction represents the set of values observed for a specific attribute across multiple packets sent by a single host. For instance, we can create a transaction for each source IP address, containing the set of destination ports it has targeted across all its packets in a day or month. This method allows the algorithm to identify values frequently targeted together by the same host. For example, it can uncover groups of ports that are commonly contacted together during the scanning activity of a source IP.

Type of transaction	Example
Row transaction	T1: [src_IP: x, dst_IP: y, src_port: 12345, dst_port: 80, protocol: TCP]
	T2: [src_IP: a, dst_IP: b, src_port: 54321, dst_port: 443, protocol: UDP]
Value transaction	T1: [80, 443, 8080]
	T2: [22, 25, 110, 80]

Table 6.1: Examples of Row and Value Transactions

After defining the transaction shapes, we can apply frequent pattern mining techniques to extract meaningful patterns from the network traffic data. In Section (2.5.1) we introduced two main techniques: frequent itemset mining and association rule mining. While the former focuses on identifying sets of items frequently

occurring together, tha latter adds directional relationships between items, providing an additional layer of insight. Table 6.2 presents examples of both association rules and frequent itemsets extracted from our dataset using the transaction types described above.

Rules	Support (%)	Confidence	Lift
On row transactions			
Antecedent: `tos=164, ttl=243, tcp_win=1025, tcp_len=6, pkt_len=44` **Consequent:** `src_hostname=62.162.214.35.bc.googleusercontent.com`	2.13	100.00	46.13
Antecedent: `ttl=49, tcp_win=42340, tos=8, tcp_len=10, pkt_len=60` **Consequent:** `src_hostname=unused-space.coop.net`	2.18	99.96	38.18
Antecedent: `tcp_win=42340, tos=8, tcp_len=10, pkt_len=60` **Consequent:** `ttl=49`	2.18	99.79	28.47
On value transactions			
Antecedent: `port=22, port=8443, port=8080, port=8888, port=3389, port=443` **Consequent:** `port=44818`	0.81	80.64	74.66
Antecedent: `port=22, port=993, port=8443, port=3389, port=8000, port=443` **Consequent:** `port=44818`	0.82	80.53	74.55
Antecedent: `port=22, port=8443, port=8080, port=9000, port=3389, port=443` **Consequent:** `port=44818`	0.81	80.35	74.39

(a) Association Rules

Frequent Itemset	Support (%)
On value transactions	
`port=23, port=80`	2.99
`port=80, port=443`	2.72
`port=8080, port=80`	2.60
`port=8443, port=443`	2.54
`port=8080, port=23`	2.41
On row transactions	
`ttl=242, tcp_len=6, pkt_len=44, tcp_win=65535`	17.88
`ttl=241, tcp_len=6, pkt_len=44, tcp_win=65535`	8.60
`src_hostname=recyber.net, ttl=239, tcp_win=1024, pkt_len=40, tcp_len=5`	5.48
`tcp_win=42340, tcp_len=10, pkt_len=60`	4.76
`tcp_len=8, pkt_len=52, tcp_win=65535`	4.20

(b) Frequent Itemsets

Table 6.2: Examples of (a) Association Rules and (b) Frequent Itemsets

6.1.3 Choosing the best approach

In the previous section we discussed about the fundamentals of frequent pattern mining such as the types of transactions and the mining techniques. To proceed with the analysis, we evaluated the combination of transaction types and mining techniques to determine the most suitable approach for our objectives. Our primary goal is to identify patterns of co-occurring items that indicate the presence of frequent types of activity. Then, we aim to track the evolution of these patterns over time. In the following sections, we present our implementation choices and the rationale behind them.

Row transactions vs value transactions

Row transactions can uncover the activity of specific hosts based on the characteristics of the packets they send. However, they are often biased by individual hosts that generate large volumes of traffic, which skews the results. Figure 6.2a demonstrates this issue, showing rules with very high confidence that involve items specific to individual hosts. Additionally, treating numerical features such as destination port, packet length, and TCP window size as categorical variables leads to an explosion in the number of unique items, complicating the analysis. Our experiments showed that row transactions often produced results that were either too generic or overly specific to individual hosts. Moreover, we might have rules without the presence of sources/destinations in the antecedent or in the consequent, creating uninformative results.

For this reasons, we excluded the use of row transactions, and shifter our attentions to value transactions. Specifically, we chose destination ports as their primary attribute. This decision is motivated by the fact that destination ports are a key indicator of the services being targeted in network traffic, particularly in the context of scanning activities. As such, they represent the primary focus for uncovering meaningful patterns.

Frequent itemsets vs association rules

After determining the type of transactions to use, we evaluated the frequent pattern mining technique that best suited our analysis.

Association rules and frequent itemsets share the common goal of identifying sets of items that frequently occur together in transactions. However, association rules go a step further by providing information about the directional relationships between items, showing how the presence of certain items (the antecedent) implies the presence of others (the consequent). While this additional information can be useful in some contexts, we determined it was not necessary for our analysis. Our primary objective is to identify sets of destination ports that are frequently targeted together, without requiring the directional relationships provided by association rules.

Frequent itemsets are better suited for our analysis, as they reveal groups of ports commonly scanned together in a simpler and more direct manner. Moreover, unlike association rules, frequent itemsets avoid redundancy issues, where multiple rules can represent the same underlying pattern with items appearing in both the antecedent and consequent. This simplicity makes frequent itemsets easier to interpret and analyze, providing a clear and concise representation of co-occurring items without the added complexity of rule directionality.

Even frequent itemsets can pose challenges, particularly when working with large datasets. The number of frequent itemsets can become overwhelming, even

when using reasonable support thresholds. This redundancy often arises because frequent itemsets can be subsets of larger frequent itemsets. To address this issue, we focused on *maximal frequent itemsets*. Maximal frequent itemsets represent the largest itemsets that are not subsets of any other frequent itemset. By focusing on these, we significantly reduce the number of itemsets to analyze while still capturing the most meaningful and comprehensive patterns in the data. From now on, for simplicity, we will continue to refer to maximal frequent itemsets simply as frequent itemsets.

6.1.4 Towards analysis

In the previous sections, we outlined the algorithms, techniques, and the scope of our analysis. Now, we present our approach for managing data and extracting frequent itemsets from network traffic.

Time resolution

A key consideration in this process is the time resolution used to construct transactions. Using long time periods, such as weeks or months, can produce overly dispersed results, as the transactions would include ports contacted over an extended time span. This approach risks correlating events that are not necessarily related. Conversely, using shorter time bins, such as hours or days, is more likely to capture events that are temporally related. Ideally, a transaction would represent the set of ports contacted by a host during a single scanning activity, which might last one or more hours. However, using very short time bins would result in a large number of transactions, significantly increasing computational complexity. To balance these considerations, we chose to construct transactions on a daily basis.

Fine-tuning parameters

Network traffic data captures activity traces from a diverse range of hosts. While some hosts are highly active, sending thousands of packets, others exhibit minimal activity, sending only a few packets. Including hosts with very low activity can introduce noise into the analysis, as these hosts may not provide sufficient data to contribute meaningfully to the identification of frequent itemsets. Additionally, they can increase variability in the transactions, biasing the results and making it harder to identify consistent patterns.

To address this issue, we applied several filtering measures to exclude irrelevant or noisy traffic before composing transactions and executing the algorithm. First, we excluded all ICMP traffic, as it does not involve destination ports and is therefore not applicable to our analysis. Second, we filtered out sources with fewer than 10 flows in the entire dataset, as their activity is not significant enough to impact the

results. Finally, during transaction construction, we optionally applied a filter to exclude ports contacted fewer than a specified number of times (e.g., 2, with 1 as default) by a sender. This additional filter further reduces noise by excluding ports that do not represent the typical behavior of a sender.

When applying frequent pattern mining algorithms, it is essential to set appropriate parameters to ensure meaningful results. Among these, the most critical parameter for generating frequent itemsets is the *minsup* (minimum support) threshold. Fine-tuning this threshold is crucial, as it directly influences the number of itemsets generated. A low *minsup* value may result in an overwhelming number of itemsets, many of which may be irrelevant. Conversely, a high *minsup* value may produce too few itemsets, potentially failing to capture meaningful patterns. The optimal *minsup* value might depend on the total number of transactions and the heterogeneity of the dataset, which refers to the diversity of observed patterns and the variability in source behavior. For highly heterogeneous datasets, a lower *minsup* value is often necessary to capture meaningful patterns, while more homogeneous datasets may allow for a higher *minsup* value.

In our study, we constructed transactions on a daily basis, requiring the algorithm to be executed separately for each day. This approach introduces variability in the number of transactions and the heterogeneity of traffic across different days. While dynamically adjusting the *minsup* value for each day might seem ideal, we opted to use a fixed *minsup* value across all days. This decision ensures consistency in the analysis and allows for straightforward comparisons of results across different days. We carefully selected the *minsup* value to ensure that, whenever possible, each day produces at least one frequent itemset, balancing the need for meaningful patterns with computational efficiency.

Daily frequent itemset mining

At this point, we show in Algorithm 4 how we construct daily transactions and apply the *FP-Growth* algorithm to extract frequent itemsets.

We preprocess the dataset by removing ICMP flows and sources with fewer than 10 flows. Next, we divide the data into daily transactions, where each transaction represents the set of destination ports contacted by a source IP on that day. Optionally, we filter out ports contacted fewer than a specified number of times (e.g., 2). Finally, we apply the *FP-Growth* algorithm with a fixed *minsup* value across all days, generating a list of frequent itemsets with their support values. We also compute the number of hosts matching each itemset by multiplying the support value by the total number of transactions for that day. Finally, we concatenate the results from all days into a single output table, which includes all day itemsets, their support, and the corresponding day with the number of matching hosts.

Algorithm 4 Daily Frequent Itemset Mining using FP-Growth

Require: `Dataset`, `min_sup`, `min_count`

function CreateTransactions(Daily flows `D`, `min_count`)
  $\mathtt{T_s} \leftarrow [\,]$ ▷ Initialize empty list of all transactions
  for each `src_IP` in `D` **do**
    `port list` ← ports contacted by `src_IP`
    `port list` ← Exclude ports with fewer than `min_count` occurrences ▷ Optionally 1 or 2
    $\mathtt{T_{src\_IP}} \leftarrow$ `port list` avoiding duplicates ▷ Single transaction for current source IP
    Append $\mathtt{T_{src\_IP}}$ to $\mathtt{T_s}$
  end for
  return $\mathtt{T_s}$ ▷ List of all transactions for the day
end function

function MineDailyItemsets(`T`, `min_sup`)
  $\mathtt{I}\langle \mathtt{itemsets}, \mathtt{supp} \rangle \leftarrow$ *FP-Growth*(T, `min_sup`) ▷ Get pairs of (`itemset`, `support`)
  for each $(\mathtt{itemset}, \mathtt{supp}) \in \mathtt{I}$ **do**
    $\mathtt{matching\_hosts} \leftarrow len(\mathtt{T}) \times \mathtt{supp}$
    Append `matching_hosts` to (`itemset`, `supp`)
  end for
  return $\mathtt{I}\langle \mathtt{itemsets}, \mathtt{supp}, \mathtt{matching\_hosts} \rangle$
end function

Step 1: Setup
Remove ICMP flows (destination port = `NULL`)
Exclude sources with fewer than 10 flows
Divide the dataset on a daily basis
Initialize empty list $\mathtt{R} \leftarrow [\,]$ to collect daily itemsets

Step 2: Construct transactions and mine itemsets
for each day `d` in `Dataset` **do**
  $\mathtt{D}_d \leftarrow$ flows for current day `d` ▷ Daily dataset
  $\mathtt{T}_d \leftarrow$ CreateTransactions($\mathtt{D}_d$, `min_count`) ▷ Daily transaction per IP
  $\mathtt{I}_d \leftarrow$ MineDailyItemsets($\mathtt{T}_d$, `min_sup`) ▷ Get *maximal* frequent itemsets
  Append all rows of $\mathtt{I}_d$ to R ▷ Concatenate all daily itemsets
end for

Step 3: Output
Output: $\mathtt{R}\langle \mathtt{Itemset}, \mathtt{Support}, \mathtt{matching\_hosts}, \mathtt{Day} \rangle$

6.2 Frequent itemsets analysis

In this section, we present the results of applying the frequent itemset mining approach outlined in the previous section to analyze network traffic data. Using the output of Algorithm 4, we extracted a comprehensive list of frequent itemsets from daily transactions, along with their support values and the number of matching hosts for each day. We present the results as heatmaps, where the x-axis represents the days of the observation period, and the y-axis lists the unique frequent itemsets identified over the entire period. The color intensity of each cell indicates the number of hosts matching the corresponding itemset on a specific day. To enhance readability, we sorted the itemsets on the y-axis based on their order of appearance. This visualization enables us to observe when specific patterns emerge, how long they persist, and their relative prevalence over time. However, it is important to

note that when generating an high number of frequent itemsets, the y-axis may not display ticks for all itemsets due to space constraints. This, however, does not affect the analysis for determining the presence and frequency of patterns over time.

We divide the analysis into two parts: first, we highlight general trends and patterns from overall internet scanning activity. Second, we conduct a focused analysis of specific groups of hosts to uncover insights into their behavior.

6.2.1 Overall internet scanning characterization

In this section, we provide a characterization of internet scanning activity by analyzing the frequent itemsets extracted from the network traffic data. We focus the analysis on the specific targeted deployments: Darknet, Layer 4 responder, and Honeypot. This approach allows us to highlight the unique patterns and trends associated with each deployment type. The analysis covers the entire observation period, enabling to observe how pattern emerges over time and their relative prevalence. As we discussed in Section (6.1.4), selecting an appropriate *minsup* threshold is essential for producing meaningful results, and this choice depends on the heterogeneity of the dataset. Given that we derive transactions from the diverse sources available on the internet, extracting frequent itemsets using a high *minsup* value can be challenging. To ensure that we capture a sufficient number of frequent itemsets for each day and deployment, we have set the *minsup* threshold to 0.5% uniformly across all deployments.

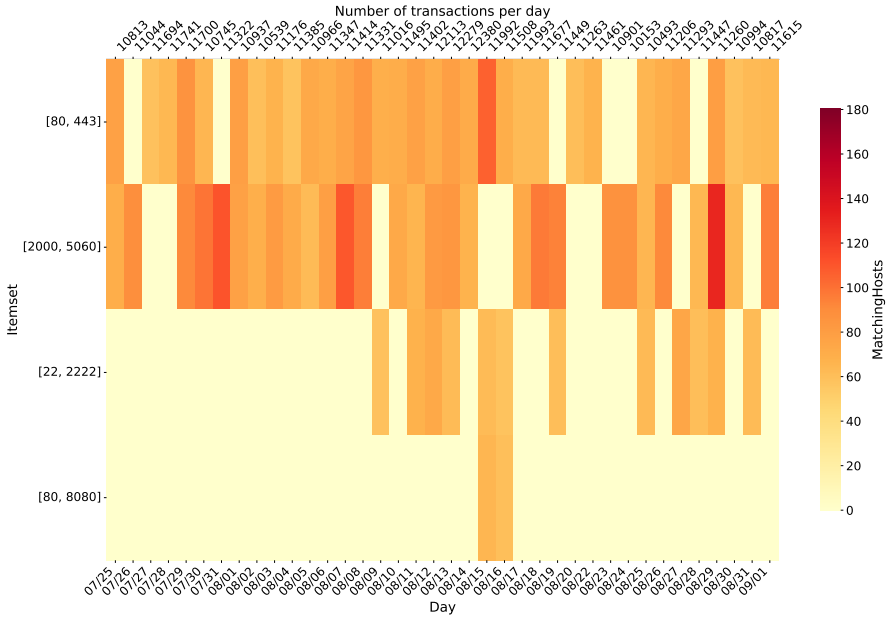
Darknet frequent itemsets

We examine the frequent itemsets extracted from both the campus and cloud darknets. As we did in Section (5.2), we consider sample of /28 subnets from each deployment to ensure a fair comparison.

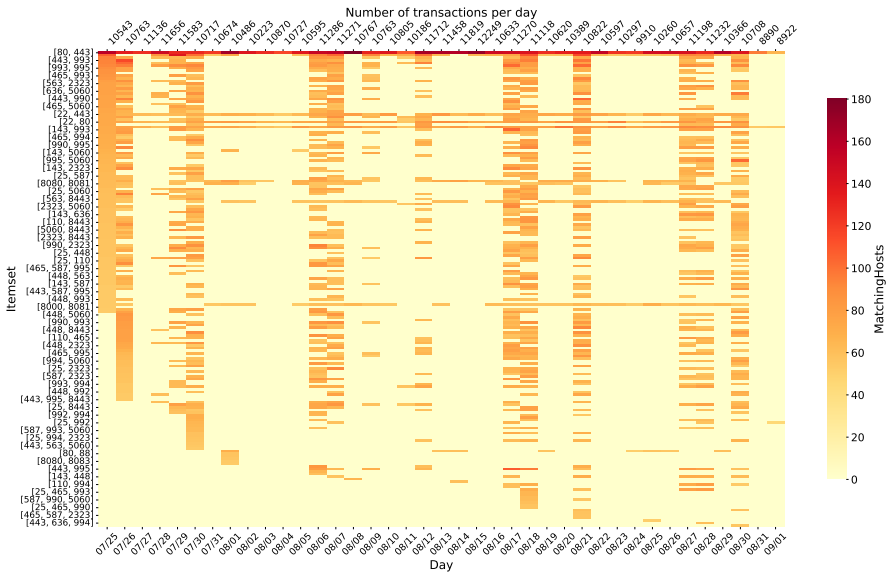
Figures 6.1a and 6.1b present the heatmaps of frequent itemsets extracted from the campus and cloud darknets, respectively. Under the same *minsup* condition and with a comparable number of transactions per day, the cloud darknet exhibits a significantly higher number of frequent itemsets compared to the campus darknet. Over the entire observation period, the campus darknet reveals only 4 frequent itemsets, while the cloud darknet identifies up to 185. This discrepancy suggests that senders targeting the cloud darknet exhibit more consistent and structured patterns of activity compared to those targeting the campus darknet.

The frequent itemsets observed in the campus darknet primarily consist of commonly targeted ports, such as {80, 443}, {2000, 5060}, {22, 2222}, and {80, 8080}, with the first two appearing consistently from the first day of observation. The number of hosts matching these patterns fluctuates between 80 and 120 per

day, indicating relatively stable but limited activity. Other patterns, such as {22, 2222} and 80, 8080, emerge after 15 and 20 days, respectively, but with a lower number of matching hosts.



(a) Frequent itemsets from campus darknet



(b) Frequent itemsets from cloud darknet

Figure 6.1: Frequent itemsets from campus and cloud darknets

The cloud darknet, instead, demonstrates a broader range of frequent itemsets from the first day of observation. Similar to the campus darknet, the most frequently observed itemsets include {80, 443}, but with a significantly higher number of matching hosts, ranging from 125 to 180 per day. Notably, while certain groups of ports are contacted persistently over time, many others appear with periodicity.

Layer 4 responder frequent itemsets

Now, we analyze the frequent itemsets extracted from the Layer 4 responder deployments. Figures 6.2a and 6.2b show the heatmaps of frequent itemsets for the campus and cloud Layer 4 responders, respectively.

While the darknet deployments in Figure 6.1 show a consistent presence of certain itemsets over time, the Layer 4 responder deployments exhibit a more dynamic pattern due to their interaction with incoming traffic during specific time windows. From Figure 6.2b, we observe that the general pattern of activity in the first phase time window is similar to what we observe in Figure 6.1b. However, as we transition into the second phase time window (July 5th - July 15th), a significant number of new frequent itemsets emerge, accompanied by an increase in the number of matching hosts. Notably, many of these new frequent itemsets emerge during the initial days of the second phase but gradually disappear over time. This behavior suggests that scanning actors initially explore the newly opened ports but later shift their focus to other targets upon realizing that the Layer 4 responder does not expose additional access points or vulnerabilities. Interestingly, during the third phase time window (July 25 - August 25), when the Layer 4 responder is disabled, the same frequent itemsets observed in the cloud darknet reappear with a similar frequency as before. This observation highlights how scanning actors heavily base their scanning patterns on the type of deployment they target. When they realize to be interacting with a Layer 4 responder, they unlock new scanning patterns.

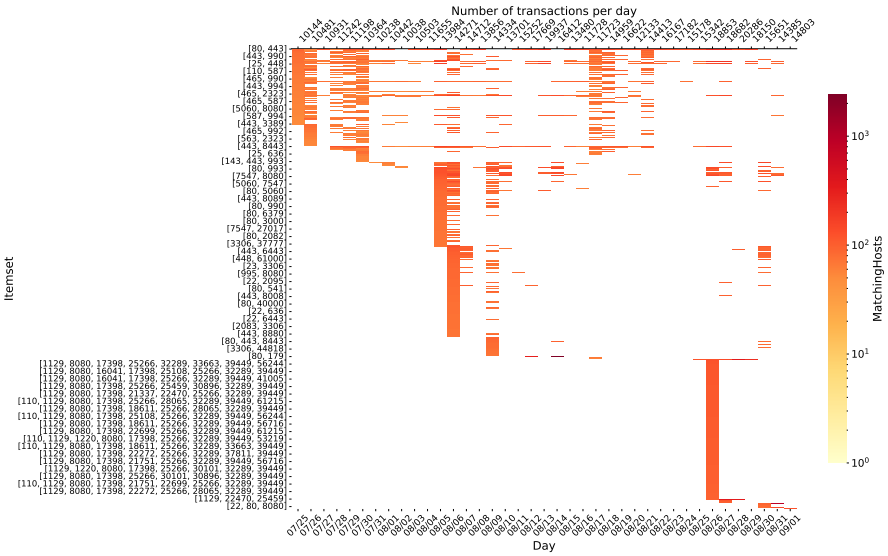
During the fourth phase (August 25 - September 2), when the Layer 4 responder is re-enabled, frequent itemsets from the second phase reappear alongside new itemsets. These new itemsets feature a larger number of ports per set, indicating broader scanning activity, observed in both campus and cloud deployments.

Honeypot frequent itemsets

Finally, we analyze the frequent itemsets extracted from the Honeypot deployment. By design, honeypot deployments are the most similar to darknet deployments in terms of the number of exposed ports.

As a result, we expect similar patterns of activity, primarily focusing on the ports that the honeypot is configured to respond to (22, 80, 443). However, to identify patterns that are more specific to honeypot deployments, we applied the mining algorithm to transactions constructed by filtering out ports contacted fewer

(a) Frequent itemsets from campus Layer 4 responder

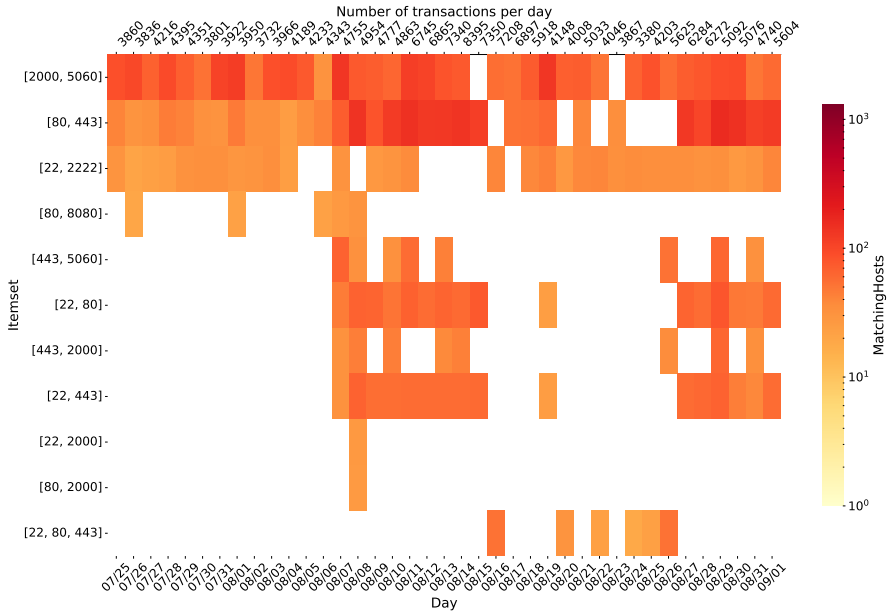


(b) Frequent itemsets from cloud Layer 4 responder

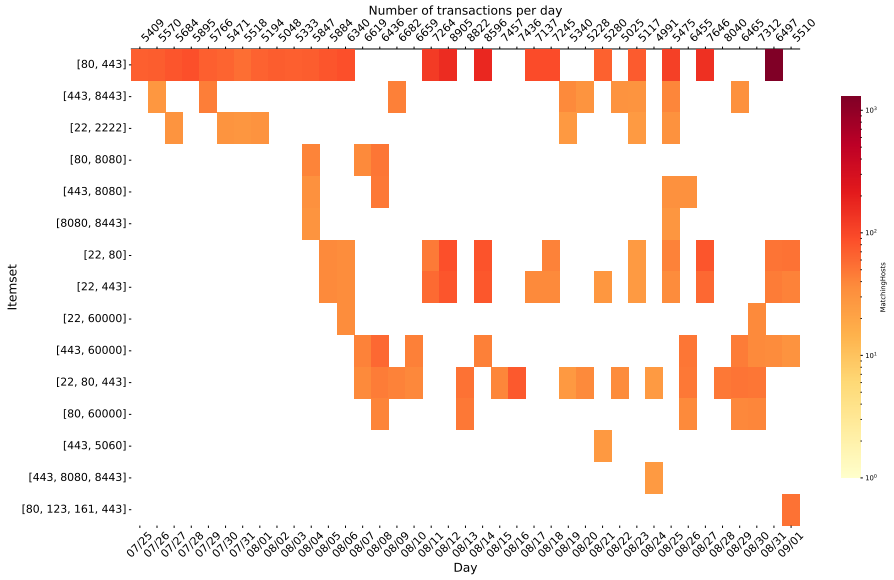
Figure 6.2: Frequent itemsets from campus and cloud Layer 4 responders

than two times by the same sender. This filtering step effectively reduces noise from darknet-like activity, which was particularly evident in the cloud darknet (Figure 6.1b). For consistency, we applied the same filtering criteria on both campus and cloud deployments.

Figures 6.3a and 6.3b present the heatmaps of frequent itemsets extracted from the campus and cloud honeypots, respectively. In both cases, the most common frequent itemsets observed until July 5th (the start of the second phase) are those



(a) Frequent itemsets from campus Honeypot



(b) Frequent itemsets from cloud Honeypot

Figure 6.3: Frequent itemsets from campus and cloud Honeypots

already encountered in Section (6.2.1), such as {80, 443}, {22, 2222}, and {2000, 5060}. Interestingly, during the interaction phases, scanners generate new itemsets that involve ports 22, 80, and 443, the ports the honeypot responds to. These

patterns disappear when the honeypot is turned off but reappear when it is re-enabled. However, on the cloud honeypot, these patterns persist even during the deactivation phase. This behavior can be attributed to the cloud configuration, where certain factors such as DNS announcements, certificate emissions, and ICMP responses remained active (see Section 5.1).

6.2.2 Hosts patterns characterization

In the previous section, we characterized overall internet scanning activity by analyzing frequent itemsets extracted from traffic targeting various deployment types. Due to the high variability and heterogeneity of the traffic, we focused on identifying general trends and patterns, using a relatively small *minsup* threshold of 0.5%. In this section, we shift our focus to extracting patterns from a more homogeneous subset of traffic by analyzing specific well known groups of hosts. By isolating and examining the behavior of hosts that share similar characteristics and are expected to exhibit comparable activity patterns, we can apply a higher *minsup* threshold. This targeted approach allows us to uncover more representative and meaningful patterns of activity, providing insights into the behavior of such specific groups.

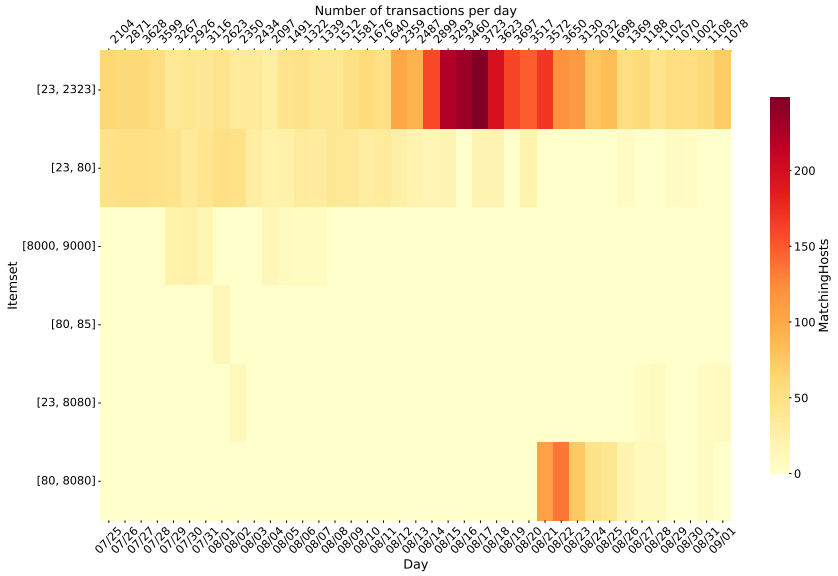
The following analysis highlights the distinct scanning patterns exhibited by homogeneous groups of hosts, targeting the entire subnet deployments (i.e., campus and cloud). For instance, we identified the activity of well-known botnets, such as Mirai [11], by detecting hosts that match a specific fingerprint (see Section 2.2.1) in their scanning activity. Beyond Mirai, we also analyzed the behavior of hosts associated with prominent scanning platforms, including Census, BinaryEdge, HiNet, Stretchoid, and Shadowserver. To classify these hosts, we performed DNS reverse lookups on their source IP addresses to resolve their hostnames, enabling us to associate their activity with specific platforms.

We group the identified hosts based on their exhibited patterns. First, we analyze cases where a single, predominant frequent itemset characterizes the activity of a group of hosts over time. Next, we explore scenarios where diverse frequent itemsets emerge in parallel each day. Then, we examine hosts that frequently change their scanning behavior. Finally, we analyze a case where a group of hosts demonstrates memory of past scanning activity.

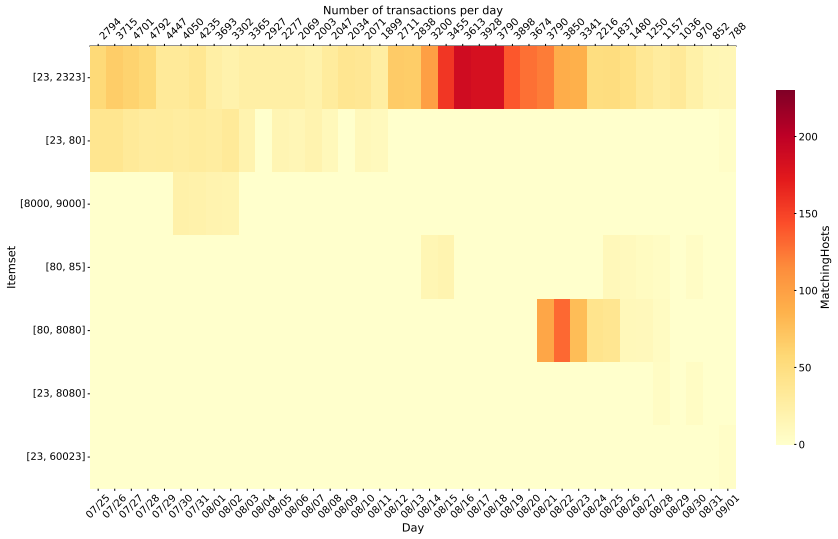
Predominant patterns

In this section, we provide an example of a group of hosts exhibiting predominant patterns over time. Figures 6.4a and 6.4b present the frequent itemsets extracted from Mirai-like hosts in the campus and cloud deployments, respectively. In both cases, the most prevalent frequent itemset consists of ports {23, 2323}, which are

the primary targets of Mirai for propagation.



(a) Campus Mirai Frequent Itemsets



(b) Cloud Mirai Frequent Itemsets

Figure 6.4: Frequent itemsets from Mirai botnet activity

From August 12th to August 24th, we observe an intensification of activity in both the campus and cloud networks. This increase is reflected in a higher number of transactions per day and a greater number of matching hosts. Interestingly, both deployments also exhibit the emergence of a new frequent itemset, {80, 8080},

around August 22nd. Notably, we use a *minsup* threshold of 0.5%. On days with peak activity, this approach allows us to characterize over 200 hosts out of approximately 3,500 active hosts per day, accounting for around 5% of the total traffic. The low threshold ensures that we can capture meaningful patterns even on less active days, providing a consistent analysis across varying traffic volumes.

Another interesting case is the activity of HiNet hosts. Figure 6.5 shows the related pattern extracted from cloud deployment, using the same *minsup* threshold as *Mirai*. While HiNet exhibits the same dominant frequent itemset, [23, 2323], the strength of this pattern is notably weaker when analyzing the number of matching hosts and transactions per day. Similar to *Mirai*, we observe an intensification in the number of matching hosts between August 12th and August 24th. However, in the case of HiNet, we are able to characterize at most 25 hosts out of approximately 1,200 transactions per day, accounting for only 2% of the total traffic. This suggests that while HiNet hosts exhibit similar scanning behavior, their overall contribution to the traffic is significantly smaller compared to *Mirai*-like hosts. We omit presenting the campus deployment, as the pattern observed is similar to the cloud one.

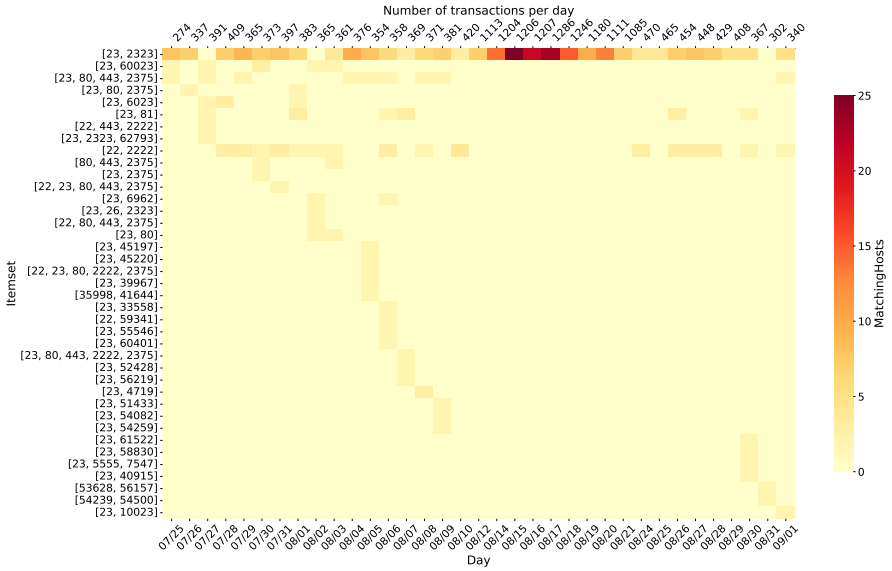
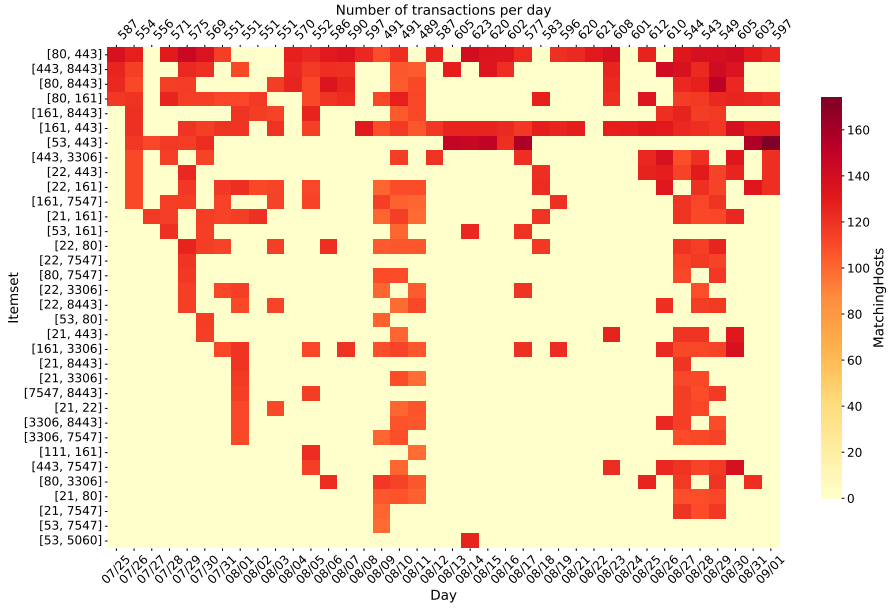


Figure 6.5: Frequent itemsets from HiNet (on Cloud)

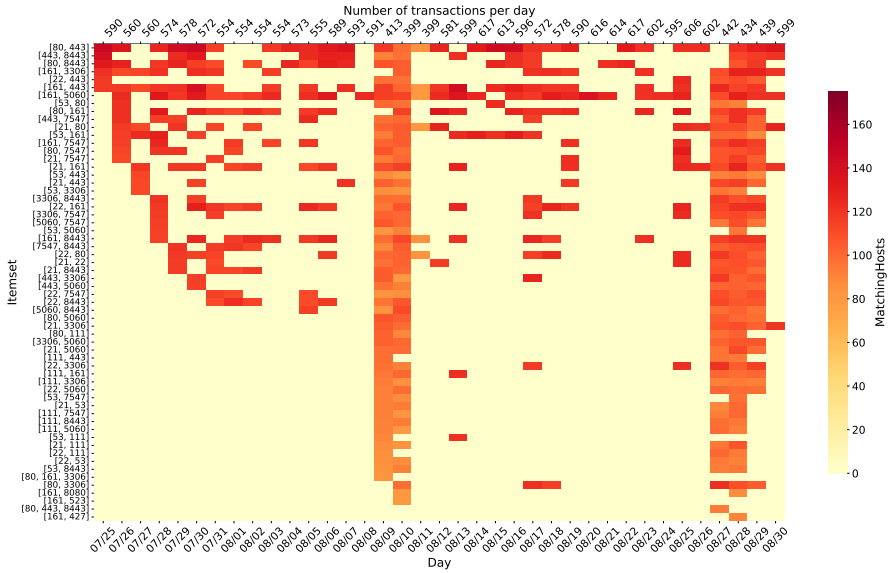
Diverse patterns

In this section, we provide an example of a group of hosts exhibiting multiple frequent itemsets each day over time. Specifically, we focus on the activity of

sources from Census project.



(a) Campus Census Frequent Itemsets



(b) Cloud Census Frequent Itemsets

Figure 6.6: Frequent itemsets from Census project activity

Figures 6.6a and 6.6b present the frequent itemsets extracted from Census project activity in the campus and cloud deployments, respectively. As for darknet deployment, from cloud we extract a higher number of frequent itemsets compared

to the campus, using an equal *minsup* of 20% and a comparable number of transactions per day. Contrary to what we observed in the previous section, where a single dominant frequent itemset characterized the activity of the entire period, the Census project exhibits a more diverse set of frequent itemsets. Specifically, on both campus and cloud networks, we observe the presence of a large presence of different frequent itemsets of ports on August 9-10th and August 27-28-29th. Specifically, we can appreciate several itemsets including ports 80, 443, 8443 and 161.

Dynamic patterns

In the previous sections we showed examples of groups of hosts exhibiting more or less the same frequent itemsets over the entire observation period, with little variation. Now, we presents examples of hosts whose frequent itemsets change day by day. Specifically, we focus on the activity of sources from Stretchoid and Binaryedge. For this analysis, we omit presenting both campus and cloud deployments, as the patterns observed are similar in both cases.

Figure 6.7 presents the frequent itemsets extracted from Stretchoid hosts' activity targeting the cloud network with a *minsup* threshold of 10%. The heatmap reveals that Stretchoid hosts consistently contact different sets of ports almost every day. This behavior suggests that these hosts perform scanning activities targeting a wide range of ports rather than focusing on a specific subset. We observe that each day, a new set of frequent itemsets emerges, while only a portion of the previous day's itemsets persist. Notably, as a new itemset emerges on a specific day, a significant number of hosts shift their activity to the new itemsets in the following days, rather than continuing to contact the same set of ports.

On the other hand, Figure 6.8 presents the frequent itemsets extracted from Binaryedge hosts' activity targeting the campus IP space, with a *minsup* threshold of 10% . Similar to Stretchoid, Binaryedge hosts also exhibit a pattern of frequently changing itemsets. However, unlike Stretchoid, where old itemsets still persist to some extent, Binaryedge hosts tend to completely abandon previous itemsets in favor of new ones. Specifically, focusing on the y-axis representing the set of frequent itemsets over time, we observe how the ports number increase as new frequent itemsets appear. This suggests how Binaryedge hosts perform sequential scans, progressively moving from lower to higher port numbers over time. From the figure, we estimate that BinaryEdge hosts take approximately 10 days to sequentially scan the first 10,000 ports. Additionally, we note an increase in the number of matching hosts starting in early August, driven by the higher number of transactions per day during this period.

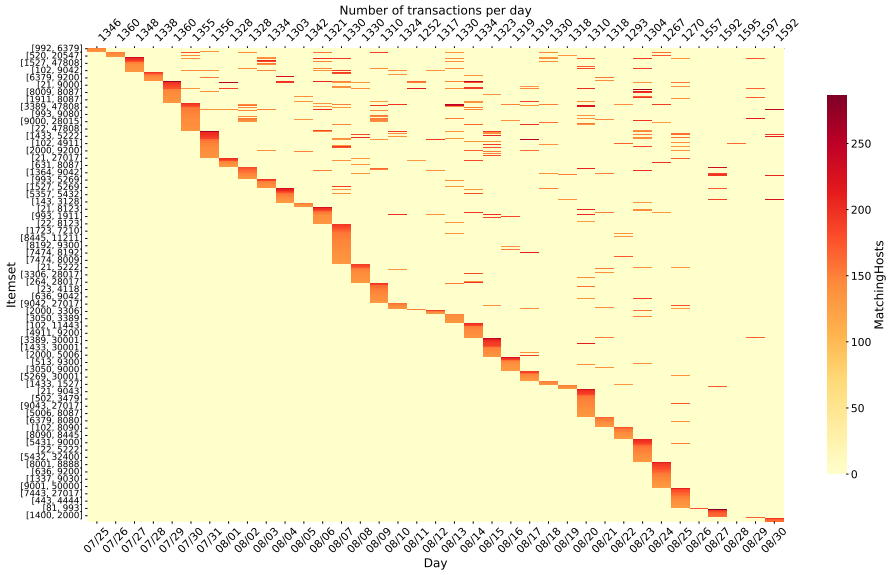


Figure 6.7: Frequent itemsets from Stretchoid (on Cloud)

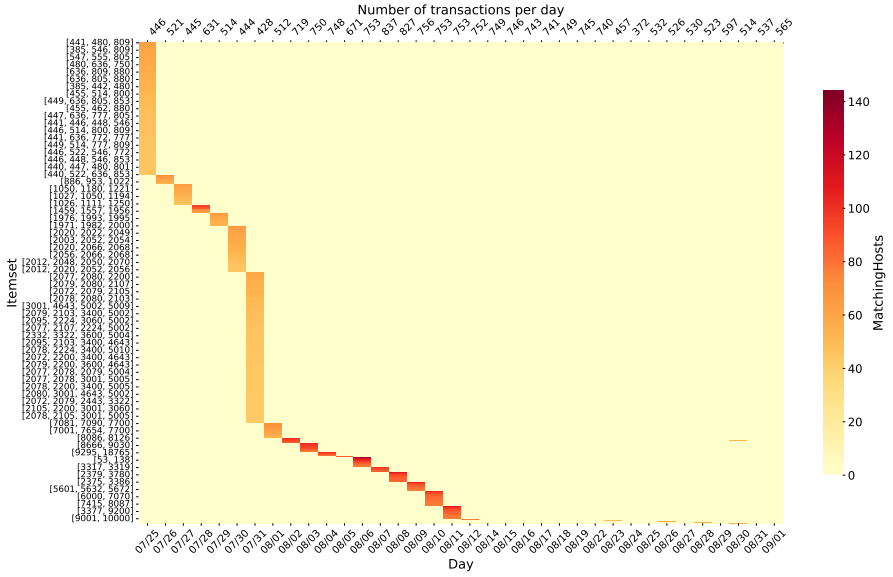


Figure 6.8: Frequent itemsets from Binaryedge (on Campus)

Hosts with memory of past activity

In this section we present an example of a group of hosts exhibiting scanning patterns showing memory of past activity. Specifically, we focus on the activity of sources from Shadowserver. For this analysis, we omit presenting both campus

and cloud deployments, as the patterns observed are similar in both cases.

Figure 6.9 presents the frequent itemsets extracted from Shadowserver hosts' activity in the campus network, with a *minsup* threshold of . The heatmap reveals that a significant portion of the involved hosts maintains a baseline of frequent itemsets, consisting of ports they contact consistently every day. These include, among others: {22, 80}, {80, 4080}, {80, 8090}, {22, 9060}, {22, 8123}, and {22, 1337}. In addition to this baseline behavior, we observe a dynamic pattern similar to what we described in Section (6.2.2), where new itemsets emerge almost daily, while older ones tend to disappear. This pattern persists until August 5th, the time of the first activation of interactive deployments. At this point, not only does a large number of new itemsets appear, but previously observed itemsets also reappear and persist over time.

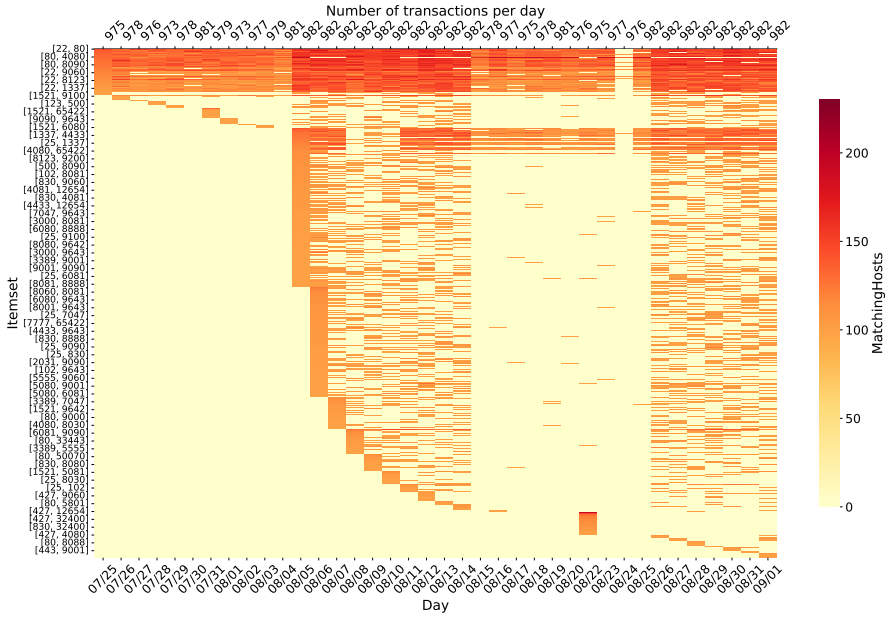


Figure 6.9: Frequent itemsets from Shadowserver (on Campus)

This behavior suggests that Shadowserver hosts systematically scan different sets of ports each day in search of open services. If no active service is detected, they shift their focus to a new set of ports the following day. However, when they identify an active host, their behavior changes significantly. They not only expand their scans to include a broader range of new itemsets but also revisit previously scanned itemsets, likely to verify whether the status of the previously closed ports has changed. Ports that they initially found as closed may now be open due to the activation of services on the targeted host, prompting the Shadowserver hosts to re-evaluate their findings.

Interestingly, the same pattern repeats in the period since the services deactivation. As for the first phase, in the third phase time window (July 25 - August 25), Shadowserver hosts return to scanning different sets of ports each day, abandoning large part of the itemset introduced in the interactive phase time window, still maintaining only a few of them. Then, in the fourth phase time window (August 25 - September 2), when the interactive deployments are re-enabled, we observe the reappearance of the same phenomenon seen in the second phase, with the reappearance of previously observed itemsets along with new ones.

6.3 Summary of results

In this chapter, we presented a comprehensive analysis of network traffic data using frequent itemset mining techniques. We began by outlining our approach to data preparation, including the construction of daily transactions and the application of the FP-Growth algorithm to extract frequent itemsets. We emphasized the importance of selecting appropriate parameters, particularly the *minsup* threshold, to ensure meaningful results. We then delved into the analysis of the extracted frequent itemsets, presenting our findings through heatmaps that illustrate the temporal dynamics of scanning activity.

Finding	**Description**
Higher frequency of itemsets in cloud darknets	Cloud darknets exhibit significantly more frequent itemsets than campus darknets (185 vs. 4), reflecting more consistent and structured scanning activity.
New patterns unlocked by interactive deployments	Interactive setups, such as Layer 4 responders and honeypots, trigger scanners to reveal many new frequent itemsets. Layer 4 responders generate itemsets involving a wide range of ports, while honeypot itemsets are primarily concentrated on the active ports it responds to (22, 80, 443).
Host with consistent predominant patterns (e.g., Mirai, HiNet)	Mirai botnet maintains stable frequent itemsets over time, primarily targeting specific propagation ports (23, 2323). Similarly, HiNet exhibits consistent patterns but with lower activity levels.
Host with diverse scanning pattern (e.g., Census)	Platforms like Census show broad scanning behavior with a wide range of frequent itemsets across many ports.
Dynamic scanning behavior (e.g., Stretchoid, Shadowserver)	Some scanners, including Stretchoid and Binaryedge, frequently shift their targets; Binaryedge appears to scan sequentially from lower to higher ports.
Scanners revisiting past targets (e.g., Shadowserver)	Groups such as Shadowserver revisit previously scanned ports upon detecting new services, indicating memory of past activity.

Table 6.3: Summary of key findings from frequent itemset analysis

Table 6.3 summarizes the key findings derived from our analysis. In a first approach, we characterized overall internet scanning activity by examining patterns observed across different deployment types, including darknets, layer 4 responders, and honeypots. However, the high variability and heterogeneity of the traffic

limited our ability to extract detailed insights. Therefore, we focused on more homogeneous groups of hosts, such as well-known botnets and scanning platforms, allowing us to uncover more specific and representative patterns of activity.

Chapter 7

Conclusions and Future Works

In this thesis, we explored the implementation of a network telescope hosted in a cloud environment, as done in several related works. While traditional network telescopes are typically deployed within university campus networks or research institutions, cloud-based telescopes monitor a set of IP addresses owned by a cloud provider. This offers several potential advantages e.g., major traffic volumes. We deployed a cloud-based network telescope using the Microsoft Azure cloud environment, and configured it with specific subnet deployments hosting fake interactive services, such as a Layer 4 responder and honeypot. We emulated a similar setup on a traditional network telescope hosted within our university campus network, and conducted a comparative analysis of the traffic received by both telescopes.

We did a statistical analysis focused on identifying differences between the traffic received by the cloud-based network telescope and that targeting a traditional telescope. We observed that the cloud-based telescope received nearly twice the volume of traffic compared to the traditional one. We then assessed the impact of deploying interactive responders, such as a Layer 4 responder and honeypot, on the volume of traffic received within specific subnets, using the port amplification factor as a quantitative measure. Our results showed that the cloud telescope experienced a higher amplification effect from the Layer 4 responder, whereas the honeypot induced a stronger amplification on the traditional telescope. Finally, we analyzed the diversity of source IP addresses contacting both telescopes. Both the campus and cloud deployments attracted a similar number of unique sources and shared many common senders. However, the cloud telescope received a higher volume of traffic from these common senders and attracted a larger number of exclusive senders—sources that contacted only the cloud telescope.

In the second part of our analysis, we applied data mining techniques to uncover patterns within the large volume of collected traffic. Using frequent pattern mining methods, we identified common patterns involving frequently contacted ports across different subnet deployments with varying levels of interactivity (Darknet, Layer 4 responder, and Honeypot). We found that traffic targeting the cloud telescope's darknet deployment exhibited a much higher number of frequent itemsets compared to the traditional telescope, indicating more consistent scanning activity aimed at the cloud network. We also examined how activating interactive responders influenced traffic patterns, uncovering new frequent itemsets particularly associated with the Layer 4 responder. The honeypot revealed frequent itemsets centered on the ports where it was active (22, 80, 443). Finally, we analyzed specific subsets of traffic generated by well-known scanning groups, including both malicious actors (e.g., Mirai) and legitimate platforms e.g., Census. Certain groups displayed predominant patterns with consistent combinations of targeted ports, while others exhibited dynamic behaviors, such as sequential scanning. Additionally, some scanners showed memory-effect behaviors, revisiting previously targeted ports to verify changes in their status.

7.1 Future Works

In this section, we list some of the possible future works.

In this work, we acquired IP addresses directly owned by the cloud provider. As a potential future direction, it would be valuable to explore IP leasing options, as discussed in Chapter 3, which typically offer more competitive pricing. After obtaining leased IP addresses, these could be advertised through the *Bring Your Own IP* (BYOIP) service offered by a cloud provider, integrating them into the their address pool. Future research could examine how traffic patterns differ between leased IP addresses and those natively owned by the cloud provider.

Another promising direction for future work is enhancing the extraction of meaningful patterns from the large volume of collected data. In our study, we applied data mining techniques—specifically frequent pattern mining—to reveal general trends and high-level behaviors of scanning groups. Building on this, exploring additional machine learning methods such as clustering and classification could provide deeper insights into the behaviors of scanning actors and help identify more specific patterns within the data.

Bibliography

[1] Ruoming Pang, Vinod Yegneswaran, Paul Barford, Vern Paxson, and Larry Peterson. «Characteristics of internet background radiation». In: *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*. IMC '04. Taormina, Sicily, Italy: Association for Computing Machinery, 2004, pp. 27–40. ISBN: 1581138210. DOI: 10.1145/1028788.1028794. URL: https://doi.org/10.1145/1028788.1028794 (cit. on pp. 1, 8, 12).

[2] Fabricio Bortoluzzi, Barry Irwin, and Carla Merkle Westphall. «Cloud Telescope: An Ephemeral, Distributed, and Cloud-Native Architecture for Collecting Internet Background Radiation». In: *IEEE Access* 13 (2025), pp. 45682–45714. DOI: 10.1109/ACCESS.2025.3549623 (cit. on pp. 4, 12, 13, 16).

[3] Eric Pauley, Paul Barford, and Patrick McDaniel. «{DScope}: A {Cloud-Native} internet telescope». In: *32nd USENIX Security Symposium (USENIX Security 23)*. 2023, pp. 5989–6006 (cit. on pp. 4, 53).

[4] Seiichi Ozawa, Tao Ban, Naoki Hashimoto, Junji Nakazato, and Jumpei Shimamura. «A study of IoT malware activities using association rule learning for darknet sensor data». In: *International Journal of Information Security* 19.1 (2020), pp. 83–92 (cit. on p. 5).

[5] Maxim Zolotykh. «Comprehensive Classification of Internet Background Noise». In: *2020 Global Smart Industry Conference (GloSIC)*. 2020, pp. 35–41. DOI: 10.1109/GloSIC50886.2020.9267850 (cit. on p. 8).

[6] David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, and Stefan Savage. «Inferring Internet denial-of-service activity». In: *ACM Trans. Comput. Syst.* 24.2 (May 2006), pp. 115–139. ISSN: 0734-2071. DOI: 10.1145/1132026.1132027. URL: https://doi.org/10.1145/1132026.1132027 (cit. on p. 8).

[7] Angela Orebaugh and Becky Pinkard. *Nmap in the enterprise: your guide to network scanning*. Elsevier, 2011 (cit. on p. 9).

[8] Nmap Project. *Nmap: the Network Mapper*. Accessed: 29 August 2025. 2025. URL: https://nmap.org/ (cit. on pp. 10, 29, 30).

[9] Harm Griffioen, Georgios Koursiounis, Georgios Smaragdakis, and Christian Doerr. «Have you syn me? characterizing ten years of internet scanning». In: *Proceedings of the 2024 ACM on Internet Measurement Conference*. 2024, pp. 149–164 (cit. on p. 10).

[10] Alberto Dainotti, Alistair King, KC Claffy, Ferdinando Papale, and Antonio Pescapé. «Analysis of a"/0" Stealth Scan from a Botnet». In: *Proceedings of the 2012 Internet Measurement Conference*. 2012, pp. 1–14 (cit. on p. 10).

[11] Manos Antonakakis et al. «Understanding the mirai botnet». In: *26th USENIX security symposium (USENIX Security 17)*. 2017, pp. 1093–1110 (cit. on pp. 10, 71).

[12] unixfreaxjp. *Mmd-0056-2016 - Linux/Mirai, how an old ELF malcode is recycled*. Accessed: 2025-08-08. 2016. URL: `http://blog.malwaremustdie.org/2016/08/mmd-0056-2016-linuxmirai-just.html` (cit. on p. 10).

[13] jgamblin. *Mirai-Source-Code*. Accessed: 29 August 2025. 2016. URL: `https://github.com/jgamblin/Mirai-Source-Code` (cit. on p. 10).

[14] Antonia Affinito, Stefania Zinno, Giovanni Stanco, Alessio Botta, and Giorgio Ventre. «The evolution of Mirai botnet scans over a six-year period». In: *Journal of Information Security and Applications* 79 (2023), p. 103629 (cit. on p. 10).

[15] Zakir Durumeric, Hudson Clark, Jeff Cody, Elliot Cubit, Matt Ellison, Liz Izhikevich, and Ariana Mirian. «Censys: A Map of Internet Hosts and Services». In: (2025) (cit. on p. 11).

[16] Internet Census Group. *Internet Census Group*. Accessed: 29 August 2025. 2024. URL: `https://www.internet-census.org/` (cit. on p. 11).

[17] BinaryEdge. *BinaryEdge – Internet-wide attack surface scanning*. Accessed: 29 August 2025. 2025. URL: `https://www.binaryedge.io` (cit. on p. 11).

[18] Chunghwa Telecom Co., Ltd. *HiNet: Internet Service Provider*. Accessed: 29 August 2025. 2025. URL: `https://www.hinet.net/globe/en/` (cit. on p. 11).

[19] Stretchoid. *Stretchoid — IP scanning platform with opt-out feature*. Accessed: 29 August 2025. 2025. URL: `https://stretchoid.com` (cit. on p. 11).

[20] Shadowserver Foundation. *Shadowserver — Global Internet Threat Monitoring*. Accessed: 29 August 2025. 2025. URL: `https://www.shadowserver.org` (cit. on p. 11).

[21] Karyn Benson, Alberto Dainotti, Kc Claffy, Alex C Snoeren, and Michael Kallitsis. «Leveraging internet background radiation for opportunistic network analysis». In: *Proceedings of the 2015 Internet Measurement Conference*. 2015, pp. 423–436 (cit. on p. 12).

[22] Eric Wustrow, Manish Karir, Michael Bailey, Farnam Jahanian, and Geoff Huston. «Internet background radiation revisited». In: *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. IMC '10. Melbourne, Australia: Association for Computing Machinery, 2010, pp. 62–74. ISBN: 9781450304832. DOI: 10.1145/1879141.1879149. URL: https://doi.org/10.1145/1879141.1879149 (cit. on p. 12).

[23] Evan Cooke, Michael Bailey, Z Morley Mao, David Watson, Farnam Jahanian, and Danny McPherson. «Toward understanding distributed blackhole placement». In: *Proceedings of the 2004 ACM workshop on Rapid malcode*. 2004, pp. 54–64 (cit. on p. 12).

[24] Vinod Yegneswaran, Paul Barford, and Dave Plonka. «On the design and use of internet sinks for network abuse monitoring». In: *International Workshop on Recent Advances in Intrusion Detection*. Springer. 2004, pp. 146–165 (cit. on p. 12).

[25] Evan Cooke, Michael Bailey, David Watson, Farnam Jahanian, and Jose Nazario. «The Internet motion sensor: A distributed global scoped Internet threat monitoring system». In: *Technical Report CSE-TR-491-04* (2004) (cit. on p. 12).

[26] D Moore, C Shannon, G Voelker, and S Savage. *Network Telescopes: Technical Report*. Tech. rep. Cooperative Association for Internet Data Analysis (CAIDA), July 2004. DOI: https://catalog.caida.org/paper/2004_tr_2004_04 (cit. on pp. 12, 14).

[27] The Tcpdump Group. *tcpdump*. https://www.tcpdump.org/. Accessed: 2025-09-04. 2025 (cit. on pp. 12, 29, 32).

[28] The Center for Applied Internet Data Analysis (CAIDA). *The UCSD Network Telescope*. https://www.caida.org/projects/network_telescope/. Accessed: 2025-09-18. n.d. (Cit. on p. 13).

[29] Barry Irwin. «Network Telescope Metrics». In: Sept. 2012 (cit. on p. 13).

[30] Peter Mell and Timothy Grance. *The NIST Definition of Cloud Computing*. en. 2011-09-28 2011. DOI: https://doi.org/10.6028/NIST.SP.800-145 (cit. on p. 14).

[31] Amazon Web Services. *AWS Documentation*. Accessed: 29 agosto 2025. 2025. URL: https://docs.aws.amazon.com/ (cit. on pp. 15, 21).

[32] Microsoft Corporation. *Microsoft Azure Documentation*. Accessed: 29 agosto 2025. 2025. URL: https://learn.microsoft.com/en-us/azure/?product=popular (cit. on pp. 15, 23, 27).

[33] Google LLC. *Google Cloud Documentation*. Accessed: 29 agosto 2025. 2025. URL: https://cloud.google.com/docs (cit. on pp. 15, 22).

[34] IBM. *IBM Cloud: Ready for AI, secure and hybrid by design*. https://www.ibm.com/cloud. Accessed: 2025-09-18. n.d. (Cit. on p. 15).

[35] Exoscale. *Cloud Pricing Models Explained: A Guide to Understanding Your Options*. https://www.exoscale.com/syslog/cloud-pricing-models/. Last updated; Accessed: 2025-09-18. n.d. (Cit. on p. 15).

[36] Fabricio Bortoluzzi, Barry Irwin, Lucas Silvero Beiler, and Carla Merkle Westphall. «Cloud Telescope: A distributed architecture for capturing Internet Background Radiation». In: *2023 IEEE 12th International Conference on Cloud Networking (CloudNet)*. 2023, pp. 77–85. DOI: 10.1109/CloudNet59005.2023.10490018 (cit. on p. 16).

[37] Mohammed J Zaki, Wagner Meira Jr, and Wagner Meira. *Data mining and machine learning: fundamental concepts and algorithms*. Cambridge University Press, 2020 (cit. on p. 17).

[38] Ben Du, Romain Fontugne, Cecilia Testart, Alex C. Snoeren, and kc claffy kc. «Sublet Your Subnet: Inferring IP Leasing in the Wild». In: *Proceedings of the 2024 ACM on Internet Measurement Conference*. IMC '24. Madrid, Spain: Association for Computing Machinery, 2024, pp. 328–336. ISBN: 9798400705922. DOI: 10.1145/3646547.3689010. URL: https://doi.org/10.1145/3646547.3689010 (cit. on p. 24).

[39] IPXO. *IPXO – Expertise-driven IP Address Solutions*. Accessed: 29 agosto 2025. 2025. URL: https://www.ipxo.com/ (cit. on p. 24).

[40] SM Huda. *ICMP Timestamp Request Remote Date Disclosure (CVE-1999-0524)*. https://wiki.smhuda.com/vulnerability-wiki/infrastructure-level/icmp-timestamp-request-remote-date-disclosure-cve-1999-0524. Accessed: 2025-09-12. n.d. (Cit. on p. 31).

[41] Internet Assigned Numbers Authority. *Telnet Options*. https://www.iana.org/assignments/telnet-options/telnet-options.xhtml. Last updated: March 16; Accessed: 2025-09-18. 2022 (cit. on p. 32).

[42] Cowrie Developers. *Cowrie SSH/Telnet Honeypot*. https://github.com/cowrie/cowrie. Accessed: 2025-09-18. n.d. (Cit. on p. 36).

[43] Dockur. *nginx-honeypot*. https://github.com/dockur/nginx-honeypot. Accessed: 2025-09-18. n.d. (Cit. on p. 36).

[44] 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: *31st USENIX Security Symposium (USENIX Security 22)*. 2022, pp. 431–448 (cit. on pp. 37, 39).

[45] 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». In: *IEEE Transactions on Network and Service Management* 20.4 (2023), pp. 5012–5025 (cit. on p. 51).

[46] Jiawei Han, Jian Pei, and Yiwen Yin. «Mining frequent patterns without candidate generation». In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. SIGMOD '00. Dallas, Texas, USA: Association for Computing Machinery, 2000, pp. 1–12. ISBN: 1581132174. DOI: 10.1145/342009.335372. URL: https://doi.org/10.1145/342009.335372 (cit. on p. 59).

[47] Christian Borgelt. *PyFIM — Frequent Itemset Mining for Python*. https://borgelt.net/pyfim.html. Accessed: 2025-09-18. n.d. (Cit. on p. 60).