Extracting Cyber Threat Intelligence from the Internet Background Noise

Thomas Mizraji

Master Thesis

Supervised by

Open Systems AG
Kirila Adamova
Andreas Jaggi

EPFL
Prof. Katerina Argyraki

March 16, 2017

Extracting Cyber Threat Intelligence from the Internet Background Noise
by Thomas Mizraji (thomas.mizraji@alumni.epfl.ch)

This master thesis was under the patronage of

Network Architecture Lab
School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne

and proudly conducted in the industry at

Open Systems AG
Räffelstrasse 29
CH-8045 Zürich
https://www.open.ch

Abstract

Following the implicit deny model for the deployed firewall policies causes that unsolicited traffic from the Internet, also known as Internet Background Radiation (IBR) or Internet Background Noise (IBN), is blocked. The goal of this research is to use the 4000 devices part of the Mission Control™ Security Services around the world as a global sensor network to monitor and analyze the Internet activities by logging, collecting and correlating blocked unsolicited traffic from the Internet.

A subset of representative hosts has been chosen to perform an analysis of today's IBN and implement a threat intelligence extraction algorithm. Besides detecting suspicious events on a single network, multiple of such events across the globe are correlated to build a confidence level for tagging and identifying the source and behavior of these events. A key to this approach is the geographical and institutional diversity of the devices. This diversity is leveraged to keep a global and unbiased view of the observed events. This information is then used to provide an overview of the current IBN trend and build real-time attack mitigation mechanisms for all protected systems based on the behavior of malicious entities.

Acknowledgements

First of all, I would like to sincerely thank the management board of Open Systems AG, as well as Josipa Kovacevic, Vice President Human Resources, for giving me the opportunity to write my master thesis within the company and experience working on real-world problems.

I would like to express my gratitude to Prof. Katerina Argyraki, head of the Network Architecture Laboratory at EPFL, for supervising my master thesis on behalf of EPFL.

My special thanks go to the Network Services teams, namely its head, Stefan Keller, as well as Andreas Jaggi, and Kirila Adamova for their seamless support regarding the ideas I came up with, the inspiring discussions we had and their feedback regarding my work. It has always been a pleasure working with such a professional team of experts. The mentoring effort provided by Kirila Adamova is worth mentioning. I am grateful for having her as a tutor and profit from her knowledge and experience. Special thanks are extended to all the employees of Open Systems AG for the interest they showed in my thesis, the advice they offered and the interesting discussions we had.

Last but certainly not least, I would like to express my gratitude to my family, for their support, encouragement and patience during my studies and for inspiring me to follow my dreams.

To infinity… and beyond ![1]

[1] Buzz Lightyear's famous catchphrase in the *Toy Story* franchise.

Contents

1

Introduction

Unsolicited traffic, also known as Internet Background Noise (IBN), has become an essential part of today's Internet traffic. Such traffic is often due to maliciously behaving machines performing port scanning in an attempt to infect other hosts by exploiting software vulnerabilities. The increasing number of connected devices also has an influence on the volume of IBN. These devices are poorly secured, thus prone to exploitable vulnerabilities. Once infected, they will also generate malicious IBN traffic trying to compromise other devices. By analyzing IBN one can get insights into current threats that could affect specific software and mitigate attacks before they happen.

1.1 Open Systems AG

Open Systems AG[1] based in Zurich, Switzerland, is a company specialized in Internet Security for more than 25 years. Open Systems AG employs more than 150 people, all focused on security, availability and visibility. As part of the Mission Control™ Security Services, Open Systems AG operates and monitors 4000 devices in 181 countries. Around 3000 of these devices are part of a distributed firewall network configured centrally. They perform firewall filtering protecting not only for traffic destined to them but also for all the traffic forwarded to the customer hosts and networks behind them.

Figure 1.1: Open Systems hosts around the world.

[1] `https://www.open.ch/`

1.2 Motivation

Following the implicit deny model for the deployed firewall policies causes that unsolicited traffic from the Internet is blocked. As this traffic has no further use for operational purposes, currently no further analysis is done on it. The goal is to use the 4000 Mission Control™ Security Services devices deployed around the world as a global sensor that will help monitor the Internet activities by logging the blocked unsolicited traffic from the Internet. Collecting and analyzing this information will give insights into the detection of suspicious events across different networks. Indeed, rather than trying to detect those events on a single network, using the correlation of multiple events across the globe to build a confidence level for tagging and identifying the source and behavior of those events. This information can then later be used to build real-time attack mitigation mechanisms for all protected systems. The thesis is about correlating the traffic and detecting malicious sources, but it also gives an updated overview of IBN.

1.3 Contribution

The contribution of this thesis can be split into three main parts.

1. **Data collection and aggregation:** Traffic data is collected from as many hosts as possible for as long as possible, collection, aggregation and storage tasks coping with realistically high load.
2. **Data correlation and analysis:** The computed statistics give a good overview of today's Internet Background Noise, they allow extracting relevant patterns paying attention to temporal differences.
3. **Information extraction algorithm:** Implementation of an algorithm that allows extracting security relevant information for the purpose of getting a detailed overview of the current IBN trends and generating blacklists of suspicious behaving Internet Protocol (IP) addresses.

1.4 Overview of the Thesis

This very chapter introduces the subject through the motivation and contributions of this thesis. Chapter 2 presents the background of the thesis covering all technical notions that are used throughout the following chapters. Chapter 3 outlines related work in the area of IBN traffic collection and Internet traffic analysis. The methodology which is used to collect traffic data is presented in Chapter 4 along with the dataset presentation and the description of the analysis tools. Chapter 5 provides an analysis of today's and past year's Internet Background Noise through different statistics and use cases validating the observations against other datasets. The Internet Background Noise information extraction algorithm is depicted in Chapter 6. Finally, future work opportunities and the conclusion of this work are given in Chapters 7 and 8 respectively.

2

Background

This chapter introduces the technical notions used in the thesis. Computer networking concepts related to network security and Internet traffic are described as well as data analysis techniques. In the first section, the concept of firewall is presented through its main components. The idea of Internet Background Noise is then introduced along with the main entities and behavior classes that cause it. Finally, data analysis and data mining techniques employed in this research are presented.

2.1 Firewall

A firewall is a combination of software and hardware components deciding if traffic is allowed in or out a local area network (LAN) or a computer [85]. A firewall can be used as barrier between a trusted network area and an untrusted one such as the Internet. Filtering decisions are based on a set of predefined rules applied to each crossing packet; packet filtering can happen at several layers. On the distributed firewall deployed by Open Systems AG, packets are filtered at the network layer, which means decisions are based on Internet Protocol (IP) addresses and port numbers.

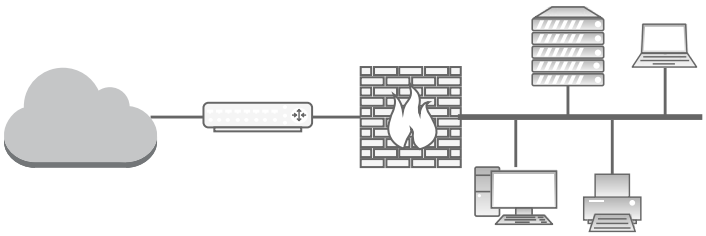


Figure 2.1: A firewall device behind a router between an untrusted network and a trusted LAN.

As stated earlier, each packet going through a firewall is only allowed to cross if it is accepted by the firewall *policy*. However, if no rule matches a specific packet, the *default policy* is applied to it. In terms of security strategies, two default policies exist. On one hand, there is the implicit deny policy [15], also called whitelist or default discard policy. It states that all not explicitly permitted traffic is blocked. This is the policy that is used for the firewalls deployed by Open Systems AG. The other possibility is the explicit deny policy, also known as blacklist or default forward policy, allowing every packet that has not been explicitly prohibited [15].

2.2 Internet Background Noise

Internet Background Noise (IBN), also known as Internet Background Radiation (IBR), denominates unsolicited one-way traffic [20, 35] going to unused/unassigned IP addresses called *Dark IPs* [87] and *darknets* [7], *network telescopes* [35], *blackhole monitors*, *network sinks* or *network motion sensors* [97] for large unused IP address blocks. In 2004, Pang et al. stated that the volume of this traffic is not minor. The incessant activity of malicious software increasing of not secure enough Internet of Things (IoT) devices makes the IBN traffic volume grow every day (see Chapter 5 and Section 5.3.1). Monitoring the IBN has proven to be a useful technique for measuring a variety of specific Internet phenomenon such as *worm* propagation [66], ongoing denial of service (DoS) attacks [67] or the impact of Internet censorship [20] (see Section 3.2.3).

2.3 Sources of Internet Background Noise

Various sources can be originator of IBN. Such traffic can be caused by computer viruses/worms and botnets, generated by hackers or be DoS backscatter. Another reason for traffic to unused IP addresses can be a software bug or misconfigurations. The topology of the different threats or traffic that compose the IBN is discussed in the following subsections.

2.3.1 Worms

A computer worm is a piece of malicious software, also known as *malware*, that spreads to other computer self-replicating and automatically exploiting vulnerabilities. Unlike computer viruses, they don't need to launch onto another program, nor need guidance from a human user. Once it is installed it can, for example, download other software such as a remote access tool to allow a hacker to take control of the compromised machine. Worms can contribute to IBN traffic in two ways: (1) trying to propagate by performing port scanning (see Section 2.3.2) (2) or performing actual attacks that generate *backscatter* traffic (see Section 2.3.4).

2.3.2 Port Scanning

As part of the network stack layered architecture, the *transport layer* lies on top of the network layer, also known as the IP layer. The transport layer provides communication services to the applications running on a computer [55]. A TCP/IP network such as the Internet provides two transport-layer protocols to applications running on host machines [55]. One of them is User Datagram Protocol (UDP); it provides a connectionless unreliable service to the invoking application. An application using UDP cares about fast retransmission and disregards packet loss. Transmission Control Protocol (TCP), on the other hand, provides a reliable and connection-oriented service. Also called transport-layer addresses, *ports* are a software abstraction that allows to differentiate specific application related packets and different machines a host is communicating with [3, 85]. TCP and UDP protocols' packet headers include *source port* and *destination port* numbers. These numbers are 16 bits unsigned integers covering the range from 0 to 65535 (2^{16}-1). Open ports, also called listening ports, are gates to the operating system to establish communication between two pieces of software on different machines. There are three classes of ports:

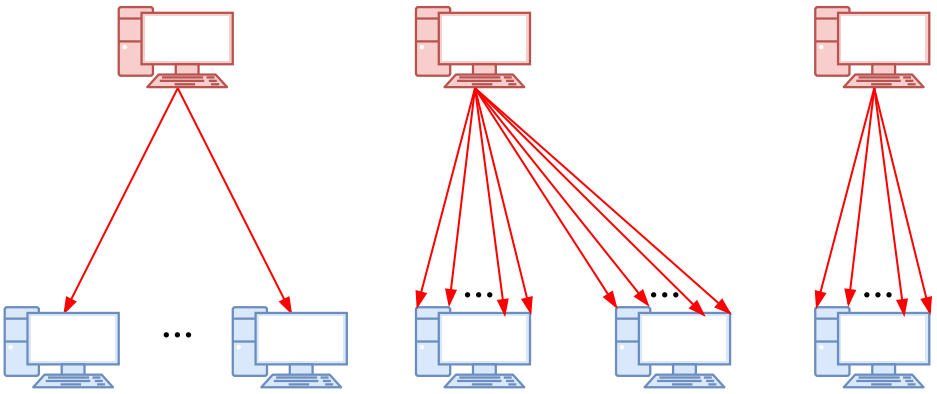
Well-known (standard) ports: 0 - 1023, assigned to services by the Internet Assigned Number Authority (IANA) [17], usually used by operating system processes

Registered ports: 1024-49151, mainly assigned by user applications

Dynamic or private ports: 49152-65535, used by user application, don't have a meaning outside an established TCP/UDP connection

If a port is listening to incoming traffic, the attached service can potentially receive malicious harmful traffic. Since every piece of software is potentially vulnerable, the ones that are bound to a specific port make no exception. Attackers are constantly trying to exploit these vulnerabilities to take over hosts. To do this, they need to know which ports are open on the target host. The technique used to identify open ports of machines is called *port scanning*. Although not being a threat per se, it is often considered as a first step of an attack since it can be a way to discover vulnerabilities.

Various techniques are used to perform port scanning [85] (see Figure 2.2). The most basic one consists in attempting to connect to all ports of a single machine to find out vulnerable software listening for some traffic. Scanning different ports of the same machine is called *vertical scanning* (see Figure 2.2c). Variations of vertical scanning include but are not limited to, focusing on specific ports, use of fragmented packets to bypass a firewall, and performing UDP scanning, which is harder because of the connectionless nature of UDP. Scanning multiple hosts on one port is also a known scanning technique called *horizontal scanning* or *port sweeping* (see Figure 2.2a). It is used to exploit universal vulnerabilities that can be shared across multiple machines because of a specific version of a piece of software. A combination of vertical and horizontal port scanning, sometimes called a *block scan* [56], is another frequent form of port scanning (see Figure 2.2b). These port scans can be performed by a single computer that is connected to the Internet. However, if these are done in a short time range, the source IP addresses are easily detected and get blacklisted. To avoid this, stealthier techniques exist to mask port scanning as usual traffic. The `FIN` scan or the `SYN/ACK` scan, for example, where the scanner sends packets with respectively the `FIN` or the `SYN/ACK` flags set. When the port is closed, a packet with a `RST` flag set will be sent, otherwise nothing is sent back. These scanning schemes, depending on filtering methods, heavy traffic, slow links and timeouts can return a lot of false positive results. This list is not exhaustive and other techniques exist such as X-Mas tree scan, Null scan, *File Transfer Protocol (FTP) bounce*, etc. [16]. The scan can be conducted in a distributed way among several machines to have different IP addresses on hand (see Section 2.3.5). The timing is also important to avoid detection, Network Intrusion Detection System (NIDS) can sometimes be evaded by adjusting the time between each probe making it slow so that it appears as random traffic [59]. Since scanning traffic is unsolicited and non-targeted, it falls into the IBN traffic category.



(a) The same destination port is probed across different machines.

(b) Combination of both schemes where the attacker scans a given set of ports on multiple destinations.

(c) A set of destination ports are tested on the same host.

Figure 2.2: Horizontal (a), Vertical (c) scans and the mixture of both called Block (b) scans. Probe packets testing for open ports are sent to potential victims. Multiple attacking machines can be used to perform port scanning in a distributed way.

2.3.3 ICMP Based Attacks

The Internet Control Message Protocol (ICMP) provides means to send error messages in several situations such as a packet not reaching its destination by sending an ICMP Type 3 `Destination Unreachable` packet [78]. It also provides a way to probe the network and gain insight into its topology [58]. ICMP can be used as a reconnaissance tool to get an overview of a target's environment, collect specific information about the target and use suitable techniques for further attacks. The most common technique, called *ICMP Sweep* or *ICMP Echo Scanning*, consists of discovering hosts within a certain IP subnet that are alive by sending *ICMP queries* and waiting for *ICMP replies*. `traceroute` [61], which makes use of ICMP, can also be employed by an attacker not only to get the route that a packet would follow but the network topology as well. `traceroute` makes use of the IP header field Time To Live (TTL) to elicit an ICMP `TIME_EXCEEDED` [78] response from each host along the path. Starting with a TTL of 1 it gradually increases it until the destination is reached[1]. Following this reasoning, *firewalking*[2] can be done to determine the filtering rules of a firewall policy [58]. Another interesting application of ICMP is Operating System (OS) fingerprinting. Depending on the form or an ICMP response, it is possible to infer the remote machine's OS and exploit specific vulnerabilities [90]. ICMP reconnaissance traffic is not solicited; thus it is part of the IBN.

[1] `https://linux.die.net/man/8/traceroute`
[2] `https://packetstormsecurity.com/UNIX/audit/firewalk/`

2.3.4 Backscatter

Backscatter traffic, in the scope of this work, is legitimate traffic generated because of unsolicited traffic. This traffic is generated by legitimate hosts in response to attacks [24] or probes by potential attackers. Two types of backscatter are discussed, namely DoS and Domain Name Service (DNS) backscatter.

2.3.4.1 Denial of Service Backscatter

The goal of a DoS attack as it can be seen in [67, 85] is to shut down an entire network or server so that it is no longer accessible. The purpose of this attack may be to inconvenience the owner of the system and cause monetary losses if the resource is commercial. A DoS attack is considered to be an easy type of attack [85]. A method to accomplish this type of attack is to flood a network with traffic so that the bandwidth of the link is consumed, and thus prevent sending or receiving legitimate data. A typical technique called *SYN flooding* is to send `SYN` packets to open a connection on the server side and leave the connection half opened to consume resources. In practice, such a packet is forged with a spoofed source IP address that is unable to respond so that it cannot end the connection. A distributed denial of service (DDoS) is a variant of the previous attack where more than one source IP address is used. Such attacks generate traffic but they also produce *backscatter* that contributes to IBN. When the packets are sent with spoofed source IP addresses, the response packets are routed to the source that has been spoofed for the request, resulting in unsolicited traffic. Packets with `RST` or `SYN/ACK` flags set may be backscatter traffic resulting from a *SYN-flood* attack (see Figure 2.3).

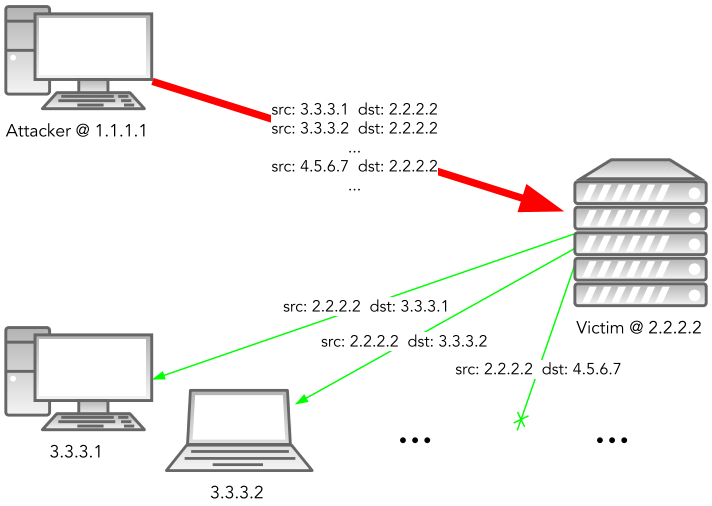


Figure 2.3: An attacker floods a server with spoofed packets. The victim responds the believed sources of these requests, thus generating backscatter traffic.

2.3.4.2 Domain Name Service Backscatter

Some machines automatically request information about remote hosts that attempt to reach them. These machines perform a reverse DNS lookup to get a domain name from a given IP address. If such a machine is under a scanning attack, it will perform at least one reverse DNS query for every remote host scanning. These queries are, by definition, backscatter, thus part of the IBN [32].

2.3.5 Botnets

A Botnet is a network composed of compromised machines called *bots* that have been previously infected by a malware, typically a computer worm. The word botnet comes from the combination of the word "robot" and "network". A botnet is commandeered by one or several botmaster(s) to achieve tasks such as DDoS, spamming, or port scanning to expand itself infecting new hosts. The botmaster takes control of the botnet via a Command and Control (C&C) channel, this way individual bots become part of the botnet and can be used to perform coordinated attacks.

Two types of architectures can be seen for botnets C&C channels. The first and predominant one is the centralized model. One C&C server, which is usually a compromised host that has access to a high bandwidth connection, is sending commands to the bots. The C&C server can be running Internet Relay Chat (IRC) or Hypertext Transfer Protocol (HTTP) to communicate. This alternative is cost-effective since only one machine is needed to control many ones, it is easy to set-up, and the latency is low, which is important for coordination purposes. A major drawback is the single point of failure. Additionally, the controller can be discovered more easily because all compromised machines communicate with one server.

The second model of C&C is the Peer-to-Peer (P2P) model that has a distributed control structure. It avoids the single point of failure issue being more resilient to network failures and it is much harder to discover and take-down because communication does not heavily depend on one host. However, given its complexity, this kind of architecture is harder to control and expand.

Most of the time, botnets are combining all aforementioned sources of IBN. The principal component of a botnet is often a worm (see Section 2.3.1) spreading across the Internet trying to infect potential victims performing port scanning (see Sections 2.3.2 and 2.3.4.2). Multiple malicious tasks can be performed by a botnet including DDoS. These attacks can induce backscatter (see Section 2.3.4.1). Therefore, botnets are IBN generators.

2.3.6 Misconfigurations and Bugs

Misconfigurations such as typing mistakes in IP addresses on a DNS or proxy server [20, 24, 72, 87] can be at the origin of unwanted traffic. Software bugs in network applications, router firmware and software [20], e.g. wrong byte ordering, may also be source of such traffic. The traffic routed to the mistyped or miscalculated address will be unsolicited one-way traffic also known as IBN from the point of view of the receiver.

2.4 Data Analysis

In this section, concepts which are used in this work for the traffic data analysis, are described. Data Analysis and Visualization techniques are outlined. The Data Science process can be summarized in five main steps [75]: (1) ask an interesting question about some phenomenon, (2) gather the data, (3) explore it, (4) model it and (5) communicate and visualize the results. The extracted aggregated results give hints in either global phenomena or local behavior classes.

2.4.1 Data Visualization

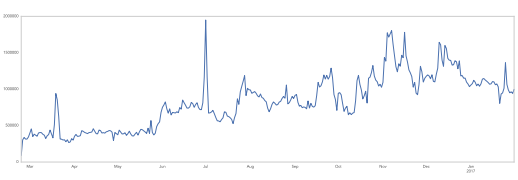
Data Visualization is a discipline consisting of the manipulation of data to get a comprehensive and human-readable display (see Figure 2.4) to bring the user a deeper understanding of its properties [6]. It allows the user to rapidly gain insight into the global trend of the data, i.e. focus on necessary information. However, some visualizations could lead to misunderstandings, on purpose or not, depending on the way they are designed [41]. The following sections describe the two main concepts that are used to represent traffic data, namely temporal (see Section 2.4.2) and spatial (see Section 2.4.3) analysis. Both visualization types share the property that observations that are close in time or space tend to be similar [89].

2.4.2 Temporal Analysis

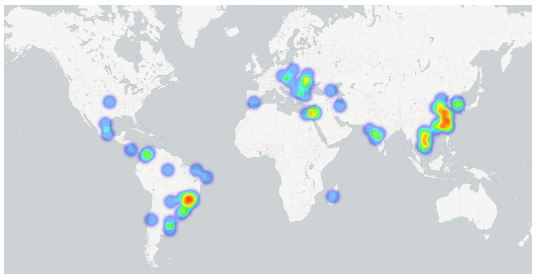
As its name suggests, Temporal Analysis is performed on temporal data, also called *time series* [89]. To detect anomalies and patterns that are part of Internet traffic, temporal analysis consisting of the number of packets received across time (see Figure 2.4a), the composition of the traffic during a specific period, the activity on some port, etc. can be used to notice *temporal differences* between time periods. Unusual traffic volume at a specific time targeting some destination port can indicate that a vulnerability has been found.

2.4.3 Spatial Analysis

Spatial data can be either areal units or points in space [89] such as geographical coordinates. Spatial analysis can be conducted on traffic data by extracting sources' locations to have an overview of the countries that generate the most amount of IBN (see Figure 2.4b). This would help gaining insight into which regions of the world are more affected by malware performing port scanning. Considering the virtual character of Internet traffic, subnets may also be seen as locations [97] where hosts that belong to the same network are most likely to share common behavior or receive the same type of traffic.



(a) Traffic volume over time



(b) Attackers Geo Heatmap

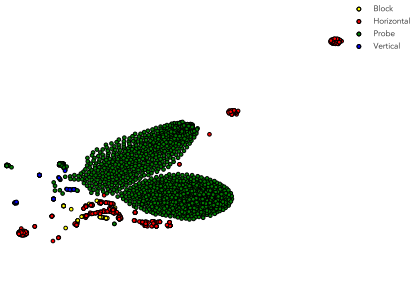
Figure 2.4: Spatial and temporal data visualizations examples

2.4.4 Data Mining

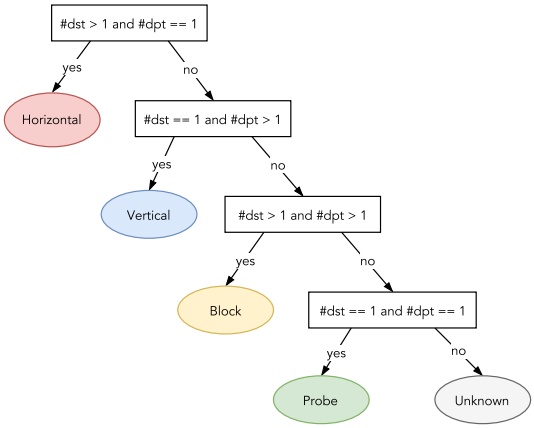
The task of extracting useful information from large datasets is called data analytics or data mining and has in the recent years become one of the most dynamically evolving areas in computer science. Data is gathered at an increasing speed and volume, with size doubling every year, but most of it is useless. The challenge is to transform tons of data into actionable intelligence e.g. from transaction data to market insights, from observational data such as satellite or sensors data to new scientific hypothesis, etc.. An example of data mining technique to get market insights, called the *market basket analysis*, tries to identify ways to boost sales by analyzing customer transaction and find frequent patterns. These patterns may take the form of items frequently bought together or items that are most likely to be sold at some specific time of the day.

2.4.4.1 Clustering and Classification

Clustering (see Figure 2.5a) is used in Machine Learning and belongs to the *unsupervised learning* approach. The basic idea is, given objects with several features/attributes that can take multiple values, find similarities between them and make groups out of these similar objects to form *clusters* [57]. A descriptive modeling technique such as clustering produces classes which are not known in advance, relying on some criteria, usually based on similarity measure, that specify when two data items probably belong to the same class. In clustering, prior knowledge on the classes/clusters to be searched for is not assumed. There exist no class label attributes to tell which classes exist. Thus, clustering serves in particular for exploratory data analysis with little or no prior knowledge. In the case of traffic flow data, possible features could be source and destination IP addresses and ports and protocol [35, 45]. Sometimes, with data objects that have too many features, *reduction* techniques have to be applied. Indeed, some clustering algorithms do not perform well with many attributes. The purpose of a reduction algorithm is to *select* and *extract* relevant features to reduce the number of attributes to a subset that still captures the meaning of the data [71]. In *Classification* (see Figure 2.5b) problems, data objects are classified/labeled. Given a database of labeled objects, a classification algorithm will attempt to build rules to classify unlabeled objects [57]. Classification belongs to the *supervised learning* algorithm category because the classes are known.



(a) Each point represents a source IP address that has been identified as a scanner during a 1-hour time window in September 2016. Features have been extracted for each source such as the number of packets the source sent, the number of different hosts, destination IPs, protocols, source and destination ports targeted by this specific source, the average number packets it sent per destination, and the rates in term of host and destination coverage. Each point is colored according to the source scanner class derived from the aforementioned features. The purpose of this plot is to discover intra class clusters, i.e. scanners from the same class that have similar observed behavior.



(b) Decision tree example that classifies scanning source IP addresses according to traffic data extracted features. (more details in Section 6.3.2)

Figure 2.5: Clustering and Classification applications

2.4.4.2 Frequent Itemsets Mining

The part of the market basket analysis used in the scope of the thesis is to find all recurring combinations among the different sets also called *transactions*. That is called *Frequent Itemset Mining (FIM)*. An itemset is considered as frequent, or recurring, if it occurs at least a fixed number of times s called the *support* value. There exist several algorithms to extract the frequent itemsets out of a transaction database. The most known one called the Apriori algorithm [2] is presented here.

IP Address	Destination Ports
1.34.62.133	[TCP/23, TCP/2323, TCP/8888]
10.250.26.108	[ICMP, TCP/23, UDP/161, TCP/10050]
104.216.4.245	[TCP/21, TCP/22, TCP/8888, TCP/8998]
104.234.162.169	[TCP/21, TCP/22]
91.98.119.156	[TCP/23, TCP/2323]
105.102.237.107	[TCP/80, TCP/21, TCP/22, TCP/8080]
110.201.64.56	[TCP/23, TCP/2323]

(a) Set of targeted ports and protocol for each source IP address

FIM

[TCP/21]
[TCP/22]
[TCP/23]
[TCP/2323]
[TCP/8888]
[TCP/23, TCP/2323]
[TCP/21, TCP/22]

(b) Extracted list of frequent destination port combinations with a support value $s = 2$

Figure 2.6: Seven Frequent Itemsets have been extracted from the transaction database which contains the ports and protocols sets targeted by each source IP address.

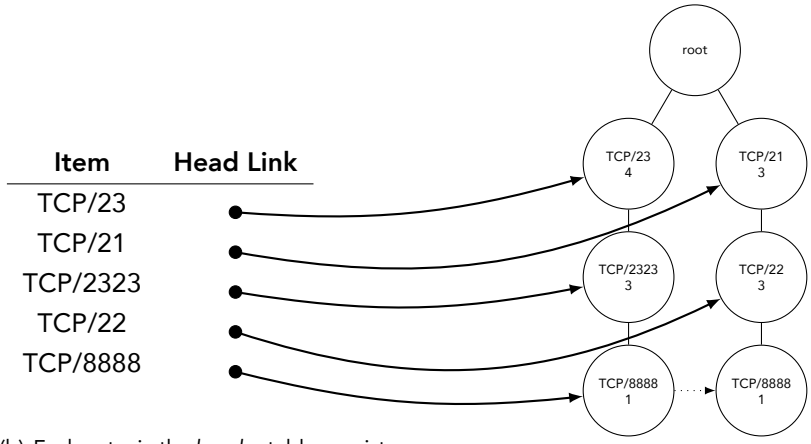
The idea behind the Apriori algorithm is going in a bottom-up approach to find the frequent itemsets. From the list of transactions T, it determines the items that occur above a given support s. The algorithm will build pairs of those items through a process called *candidates generation* or *joining step* combining only the frequent one-sized itemsets, i.e. the items that appear in more than s transactions. The next step, called the *pruning step*, consists in keeping only the pairs that occur at least s times among the transactions T. The procedure continues until no other frequent itemset is found. To be a frequent itemset, all subsets of an itemset must be frequent, thus no other frequent itemset will be found generating other larger candidate sets.

2.4.4.3 FP-Growth Algorithm

In [36], Han et al. devised the *Frequent Pattern Growth* algorithm. The method aims to mine the frequent itemsets of a database of translations without candidate generation. The process of generating candidates may lead to a huge set of potential frequent itemsets and testing them requires multiple scans of the database (see Section 2.4.4.2). To overcome this problem, [36] compress the original database into a tree structure called the *FP-Tree* (see Figure 2.7). The benefits of such a structure are its completeness, with a lossless representation of the frequent itemsets, and its compactness, thanks to a frequency sorted representation of the items [36]. The idea is then to recursively grow frequent patterns by constructing pattern conditional trees until the resulting FP-Tree is empty or it contains only a single path [36].

IP Address	Ordered Frequent Itemsets
1.34.62.133	TCP/23, TCP/2323, TCP/8888
10.250.26.108	TCP/23
104.216.4.245	TCP/21, TCP/22, TCP/8888
104.234.162.169	TCP/21, TCP/22
91.98.119.156	TCP/23, TCP/2323
105.102.237.107	TCP/21, TCP/22
110.201.64.56	TCP/23, TCP/2323

(a) Only items occurring more than the specified threshold s are kept. Each list is sorted in descending number of appearances in the transaction database (see Figure 2.6a).



(b) Each entry in the *header* table consists of two fields, (1) *item-name* and (2) *head of node-link*, that points to the first node in the FP-Tree carrying the *item-name* [36].

(c) Each node in the *item prefix tree* contains three fields, (1) *item-name*, that states which item the node represents, (2) *count*, that registers the number of transactions or source IP addresses represented by the portion of the path reaching the node, and (3) *node-links* pointing to the next node in the tree that carries the same item-name, if it exists [36].

Figure 2.7: The FP-Tree needs three steps to be built. The transaction database is scanned once to find all single frequent items. Single frequent itemsets are sorted by frequency in descending order (see Figure 2.7a). The tree is then built after a final database scan (see Figures 2.7b and 2.7c).

2.4.5 Information Theoretical Entropy

The concept of entropy is used here to measure the *observational variety* in the observed values of some feature [18, 98]. A random variable X which may take different values, is observed m times. Since m is finite an empirical probability distribution of X, $p(x_i) = m_i/m$, $x_i \in X$ can be computed, where m_i is the number of times the value x_i is observed and $p(x_i)$ is the probability that X takes the value x_i. The entropy of X is computed as:

$$H(X) := -\sum_{x_i \in X} p(x_i) \log p(x_i) \tag{2.1}$$

Through Equation (2.1), it can be noticed that the entropy of a random variable increases when its density is more uniform i.e. the values are equally frequent. To compare the entropy from features of different types, Xu et al., in [98], introduce the notion of *standardized entropy*, also referred as *Relative Uncertainty (RU)*. Considering that X may take up to N_X different values, the maximum number of values X could have taken in the sample is either N_X or m. The maximum possible entropy, as if the random variable was uniformly distributed, is computed to standardize the actual entropy value (see Equation (2.2)).

$$RU(X) := \frac{H(X)}{H_{max}(X)} = \frac{H(X)}{\log \min \{N_X, m\}} \tag{2.2}$$

Relative uncertainty provides a *uniqueness* of the measured values. As the variable distribution tends to uniformity, its RU gets closer to one. The idea is to focus on traffic coming from specific sources to compute RU of its destination ports or destination IP addresses to get a measurement of their diversity.

3

Related Work

This chapter outlines related work in Internet Background Noise detection and analysis. It starts with the monitoring phase. Next, the discussions focus on research in the field of characterization and intelligence extraction of Internet Background Noise (IBN). As the research in this area is extensive, only a selection of the work available is presented here.

3.1 Internet Background Noise Monitoring

There are different ways to monitor Internet Background Noise. This section presents some of the most used methods in the research field. Although most of the following research focuses on a deep inspection of packets, in this work the emphasis is on information retrieved from firewall dropped traffic logs. Regardless of the source, the main challenges in IBN monitoring are still the design and scalability of such a system. Rerouting traffic destined to network telescopes to a predefined infrastructure with the help of Border Gateway Protocol (BGP) announcements [72, 97], capturing packets at backbone routers [35, 87, 98], or deploying sensors/honeypots [72] on the network are ways to monitor IBN. Network telescopes (see Section 3.1.1) are a great source of traffic data, nevertheless the monitoring system must cope with the large volume of data. Virtual Sensors (see Section 3.1.1.2) and Grey Internet Protocol (IP) Networks (see Section 3.1.2) overcome this difficulty by using less sensors. Thus, it may provide less but more qualitative traffic data.

3.1.1 Network Telescopes

In [72], nonproductive traffic is monitored with characterization purpose (see Section 3.2). Their method is to use data collected from four *network telescopes*, also known as *Darknets* or *Internet Sink-holes*, within the IPv4 address space. These are large unused IP address ranges, also known as *Dark IPs*, which gather unwanted traffic destined to either nonexistent/non-running hosts or servers that are not waiting for this traffic. Two approaches are used to collect data. The first one is collecting and analyzing packets using passive measurement of background radiation, i.e. only collecting traffic without responding to it. In the other approach, an active stateless monitoring system called Active Sink[1] is implemented. Its purpose is to send the expected answers to potential attackers and simulate machines (see Section 3.1.1.1) with a stateless system. Using the stateless approach gives the platform more scalability dealing with high loads of traffic. Another passive collection of Internet Background Radiation (IBR) has been done in [7] by Benson et al. studying the nature of IBR sources, observing the noise consecutive to some threats, noting the frequency of the sources generating IBR, and correlating in the positions of IP addresses receiving IBN traffic.

[1] Active Sink: An Event-driven Stateless Responder Platform

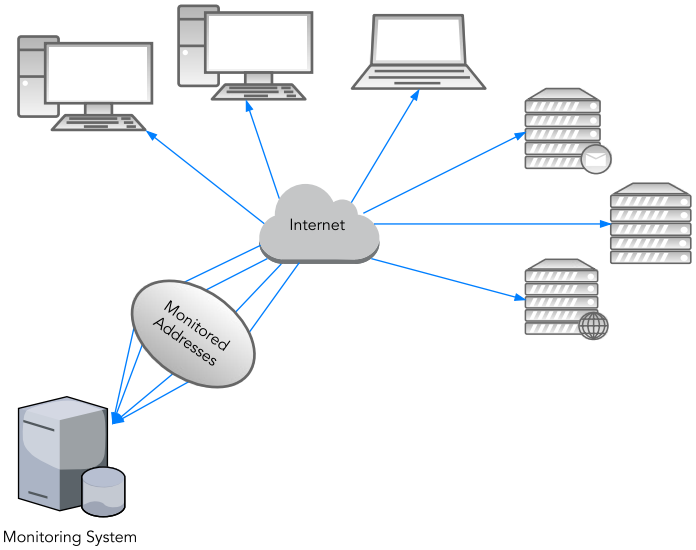


Figure 3.1: Network telescopes are globally routed and monitored networks. Since there are few or no allocated IP addresses in the monitored range, almost all traffic destined to the network is illegitimate and part of IBN.

As introduced in the previous paragraph, dealing with traffic data such as flow traces or full packet captures introduces major challenges in term of scalability. In fact, Pang et al., in [72], devise a filtering scheme balancing the trade-off between data volume reduction and information loss as the filtering process is applied. Various filtering schemes are proposed but the focus stays on the *Source-Destination Filtering*. It uses source IP addresses and relies on the assumption that a host will exhibit the same behavior on all other destination IP addresses. However, this scheme cannot detect a worm that would have, for example, different patterns depending on the destination IP and would only capture a portion of its behavior.

Continuing and updating the effort that was conducted by Pang et al., Wustrow et al. in [97] also made use of darknets to make full packet captures of IBN, using a tool based on `libpcap`[2]. To collect traffic in [97], the scientists have been allowed by the ARIN[3] and the APNIC[4] to announce unallocated network blocks to the Internet via BGP [14]. The whole dataset was split into two sets. The first was used to study spatial properties of the traffic and the second—for temporal properties, by taking the same specific week for each year from 2006 to 2010.

[2] `http://www.tcpdump.org/`
[3] `https://www.arin.net/`
[4] `https://www.apnic.net/`

3.1.1.1 Honeypots

In computing, a *honeypot* is a security mechanism set to monitor traffic and gather information about a potential attacker [27, 40, 85]. Usually honeypots are decoy servers or systems that are setup to be an easier prey than productive hosts for intruders. There are two reasons to setup such a system: (1) learn about intruders and how they attempt to gain access and privileges of systems to better protect real systems and (2) gather forensic information required to apprehend intruders. Thus, the goal is to act as a vulnerable host and gather as much information as possible from intruders [72, 97].

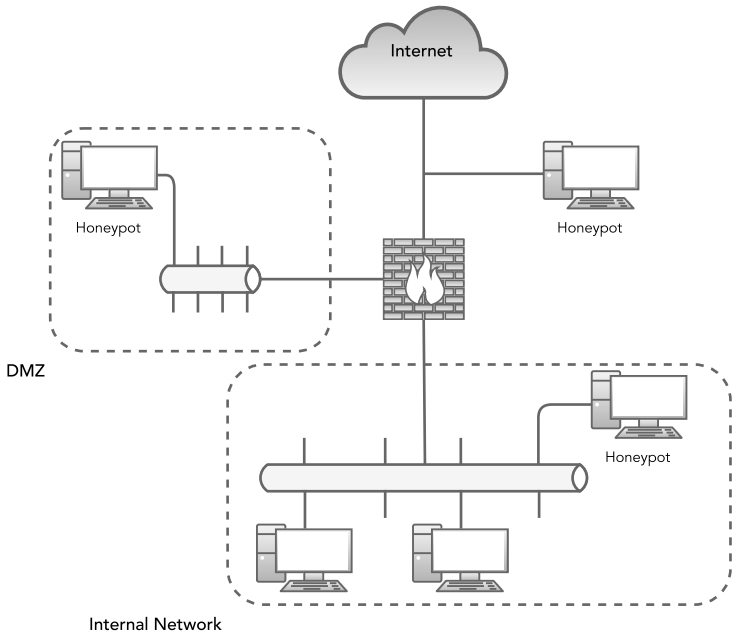


Figure 3.2: Honeypots can be setup at different places in a network topology depending on their purpose. They can be setup inside, outside or in a demilitarized zone (DMZ) of a firewall design, for control purposes [27].

3.1.1.2 Virtual Sensors

A subcategory of Darknets has been devised by Shimoda and Goto in [87] proposing the concept of *Virtual Sensors*. The idea is to locate IP addresses with the same characteristics as Dark IPs. These are machines that never send any packets from flow attributes based analysis. When addresses are classified as Dark IPs, every incoming packet is seen as an anomaly packet, malicious or not. There are some advantages in using Virtual Sensors: (1) reduced costs for not deploying physical sensors, (2) better accuracy by analyzing many sensors simultaneously and (3) the variety of traffic type that can be analyzed. Because capturing all packets is difficult, most routers export *sampling* packets. That is why the accuracy of detecting anomalous packets is degraded [87]. Besides, locating the virtual sensor among all machines is a heuristic based method, thus error-prone. A naive way to locate unused IP addresses would be to look for *one-way packets* and classify an IP address as unused if it only appears in the destination field of packets. Nevertheless, as stated by [87], not all one-way packets are malicious, e.g. SNMP trap, RIP, `netflow`, sFlow, and the naive approach of just looking at packet headers does not consider *asymmetric routes*, when the sending route is not the same as the receiving route. The paper also assumes that the fraction of non-malicious one-way packets is smaller than the portion of malicious ones. To cope with these difficulties an algorithm has been devised and is presented in [87]. It consists of multiple states and transitions source IPs

can go through to be classified as Dark IP or not. This classification process permits filtering network flows and detecting anomaly packets thanks to the Virtual Sensors IP address databases. Shimoda and Goto obtained similar data as other known sources of IBN traffic in a cost-effective way.

Glatz and Dimitropoulos, in [35], collected a large dataset of network traces (flow-level data) destined to or coming from the Internet and then identified *one-way traffic*, i.e. traffic destined to unused IP addresses. The data comes from an academic backbone network where the monitored range is about 2.2 million IP addresses. Using this type of data involves a lot of sanitization. The first step is eliminating the double counted flows. If a flow is routed through more than one border router it is reported multiple times. Merging the matching flows that have been previously fragmented by the measurement tool `netflow`[5] is the second step. Finally, the two-way matching flows are paired to end up with only one-way flows.

3.1.2 IP Gray Space

Similar to the concept of Virtual Dark IP addresses (see Section 3.1.1.2), Jin et al. present *IP Gray Space* analysis which is another innovative way to collect traces, also called flow attributes, of background traffic representative from Internet ongoing threats in [47, 48]. Motivated by the observation that within a typical corporate/academic network, owning one or several address ranges, not all IP addresses are assigned at any given time, Jin et al. conducted traffic monitoring on scattered non-assigned IP addresses of the campus network. One of the benefits of this method is that unlike traditional Darknet analysis (see Section 3.1.1), malicious entities are less likely to stay away from Gray Space. Publicly known Darknets might be avoided by attackers to keep a low profile vis-à-vis the security community. In other words, an attacker would more likely target addresses that are temporary gray because they belong to an active network and not to a Darknet. A simple heuristic algorithm to extract the IP gray space was developed in [47] saying that an address is considered member of the gray space, i.e. is *gray* or inactive, over the period $[t_0, t_0 + T]$ if and only if no traffic has originated from this address during the time range $[t_0 - \tau, t_0 + T + \tau]$, T being the time period length and τ is some fixed constant. *Greynet* activity has also been studied by Miao et al. in [65].

3.2 Internet Traffic Characterization and Intelligence Extraction

Once the data is collected, different procedures can be applied to gain insight into its composition and detect patterns that define either ongoing threats, misconfigurations or global Internet events.

3.2.1 Internet Background Noise Traffic Composition

Pang et al. did passive and active measurements of the IBR and analyzed the traffic composition based on the most frequent protocols. Most popular destination ports are extracted to help build relevant honeypot software. A temporal analysis is also done on backscatter activity looking at the Transmission Control Protocol (TCP) `SYN`, `ACK` and `RST` flags on the packet headers. Similar measurements have been made in [97] by Wustrow et al. who also performed active monitoring and inspected traffic on application level to give an overview of the trending threats and main sources of IBR in 2010. In [72, 97], traffic is broken down into protocol, destination and specific exploit thanks to full packet capture. A deeper investigation is done on popular activities, i.e. popular destination port, to find exploits or typical misconfigurations. Traffic generated by Welchia, Blaster [5], Slammer worms is extracted and analyzed. Malformed or malicious packets targeting known vulnerabilities in corporate software traffic is also observed. In [87], focus is also made on targeted destination ports to detect Internet threats through IBN analysis.

[5] `http://www.cisco.com/go/netflow`

3.2.2 Internet Traffic Profiling

Xu et al. developed a methodology to classify Internet traffic in [98] not only detecting suspicious behavior, but all types of patterns identifying the behavior of legitimate servers as well as potentially malicious scanners using traffic trace data. They combine data mining and information theoretic techniques on traffic trace data to automatically discover *significant behavior of interest* and provide an interpretation of these behavior classes. Using the well-known tuple `(srcIP, dstIP, srcPort, dstPort)` and extracting the *significant cluster of interest* along each dimension, the authors gain insight into communication patterns fixing the `srcIP` or `dstIP` dimensions. Typical communication patterns include a web server replying to a large number of hosts or scanning traffic. On the other hand, the `srcPort` and `dstPort` provide information about the aggregate ports behavior. After extracting the significant clusters for a given fixed dimension, the *relative uncertainty* [98], or *standardized entropy*, is computed for each of the remaining *free* dimensions, the ones that have not been fixed. Several behavior classes are defined given the relative uncertainties of the free dimensions. Xu et al., then proceed with the analysis of dominant states to get substantial values of each dimension and use them to build classification rules for the traffic. Intra-cluster analysis is also done to distinguish, among others, sub-classes of behavior. A similar procedure has been developed in [45] by Iglesias and Zseby using information theoretic methods coupled with machine learning tools used to cluster the different behavior classes.

In [35], Glatz and Dimitropoulos devised a procedure to extract *one-way traffic*. They designed a simple, rule-based one-way traffic classification scheme that only relies on flow-level data, unlike other schemes that do packet analysis [72, 97]. Scalability, easy configuration without training needed, comprehensive classification process and extensibility are the key points of the scheme according to the researchers. *Rules* are used to classify flows, they are sets specifying the signs a flow should or should not have [35]. The rules have been devised with the help of a refinement process that iteratively eliminates classification conflicts.

The focus of methods discussed above is mostly on traffic classification of all traffic, legitimate or not. Ertoz et al. designed MINDS [25], a Network Intrusion Detection System (NIDS) that makes use of machine learning methods to find anomalies and inspect outliers. According to the researchers, MINDS overcomes state-of-the-art signature-based tools such as SNORT[6]. The system filters the traffic data and extracts its features such as the number of flows from a specific source inside the network over the last T seconds or the number of flows to a defined destination IP address using same source port over the last N flows [12]. Distinction is made between known attacks and detected novel ones and a summary of unseen attacks is sent to a human analyst for further investigation. Data mining techniques are also used by Vaarandi and Podiņš in [94] to design an unsupervised NIDS. Applications of new generation NIDS on detecting port scanning attacks have been discussed by Treurniet in [91] as well, where a specification-based state machine is designed for classification purpose. Treurniet developed finite state machines for TCP, User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP) to classify *sessions*, i.e. two computers exchanging traffic using the same network protocol [91]. It searches for well-defined patterns such as TCP flags combinations successions or ICMP type and code value successions to group sessions into common activities called *activity patterns*. Activity patterns are tuples where the number of unique source and destination IPs are extracted along with the number of different source and destination ports.

Jung et al. developed Threshold Random Walk (TRW) in [51], an online detection algorithm that identifies malicious remote hosts from traffic traces using sequential hypothesis testing. Sequential hypothesis testing detects scanners using positive reward. For every new event, a ratio is computed between conditional probabilities that relies on predefined *scanner* or *benign* hypotheses. When a source succeeds in connecting to a host, its ratio decreases. On the contrary, its ratio increases when a connection attempt fails [68]. When the ratio goes above a fixed threshold, the source is

[6] `https://www.snort.org/`

considered as a scanner. Conversely, when the ratio goes below some threshold, the remote host is classified as benign. The upper and lower thresholds are fixed based on desired true positive and false positive rates [68]. TRW makes use of the observation that scanners are more likely to scan hosts that do not exist or do not have a service activated than legitimate hosts, since attackers often lack precise knowledge of their target.

In [12], Brownlee, with `iatmon`[7], focuses on one-way traffic classification into groups that share the same Interarrival Time (IAT) summarizing sources behavior over time. The traffic data comes from a network telescope (see Section 3.1.1). According to Brownlee, IAT is the time it takes between two consecutive packets from the same source IP address to reach the network telescope. However, in the scope of this work, the sense of IAT varies and is focused on a pair of communicating hosts instead of one host and a whole network.

3.2.3 Internet Background Noise Intelligence Extraction

Many techniques have been developed by researchers to extract various types of relevant information from traffic data. Using the classification and characterization methods, patterns such as such as port scanning [21, 35, 47, 48] can be extracted. In [21], Dainotti et al. focus on the behavior of a botnet performing Session Initiation Protocol (SIP) scanning (UDP/5060) over the whole IPv4 address space. By performing *backscatter* analysis [32, 33, 67], one can infer and provide an overview of ongoing attacks over the Internet. IBN may also be a source of information about the malware zoo activity across the Internet. The Slammer worm was studied thanks to Network Telescopes monitoring unused address spaces [66]. Paradoxically, malware activity can be profitable to gain insight into the current state of the network. Dainotti et al., in [20] studied the impact of country-level censorship in Egypt on January 27, 2011 and two earthquakes that struck New Zealand and Japan in 2011 by examining the number of unique IP addresses per hour, which contribute to IBR. Dainotti et al. are able to approximate the impact radius from an earthquake's epicenter based on the lack of radiating source, i.e. the ones that lost Internet connectivity. The same reasoning is done to study country-level outages induced by government censorship. IBN can also be used to monitor the availability of services [7, 35] and locate Domain Name Service (DNS) open resolvers that may be responsible of denial of service (DoS) attacks [7].

[7] `https://www.caida.org/tools/measurement/iatmon/`

4

Methodology

This chapter introduces the dataset used in this work, including how the data is collected and parsed. Additionally, the methodology for analyzing this data is presented.

4.1 Dataset

As introduced in the beginning of the report, the data is comprised of Internet Protocol (IP) packet headers from IPv4 traffic packets that have been denied by hosts deployed by Open Systems AG. These hosts implement firewalls that follow the implicit deny policy. Traffic that is not explicitly permitted is denied. Furthermore, most of the hosts where the data is collected are not supposed to receive traffic other than management traffic. Nearly all sensors are Management devices used as intermediates to configure internal hosts. That is why there is a need for an additional link, which is not affected by customer traffic or issues on their links, to manage the devices. These links are directly connected to the Internet, thus observing Internet Background Noise (IBN) traffic. Some hosts are dedicated firewall hosts that protect publicly accessible machines, therefore, they also have direct Internet connection. The following subsections describe the data collection process by outlining a generic customer topology, the tools that are used to filter the traffic, log it, and send it to an aggregation host. The diversity of the sensors in terms of geographic location, connectivity and IP address in the IPv4 range is also described.

4.1.1 Data Collection

This section describes from which hosts and with which tools the traffic data is gathered and aggregated among all sensors hosts. Since firewall basics are introduced in Section 2.1, more technical details are outlined in the subsequent sections.

4.1.1.1 Generic Network Topology

A typical network topology that can be encountered at customers is visualized in Figure 4.1. The topology consists of three main parts: (1) the customer local area network (LAN), (2) several demilitarized zones (DMZs) where mail or web servers can be located, and (3) the Management LAN.

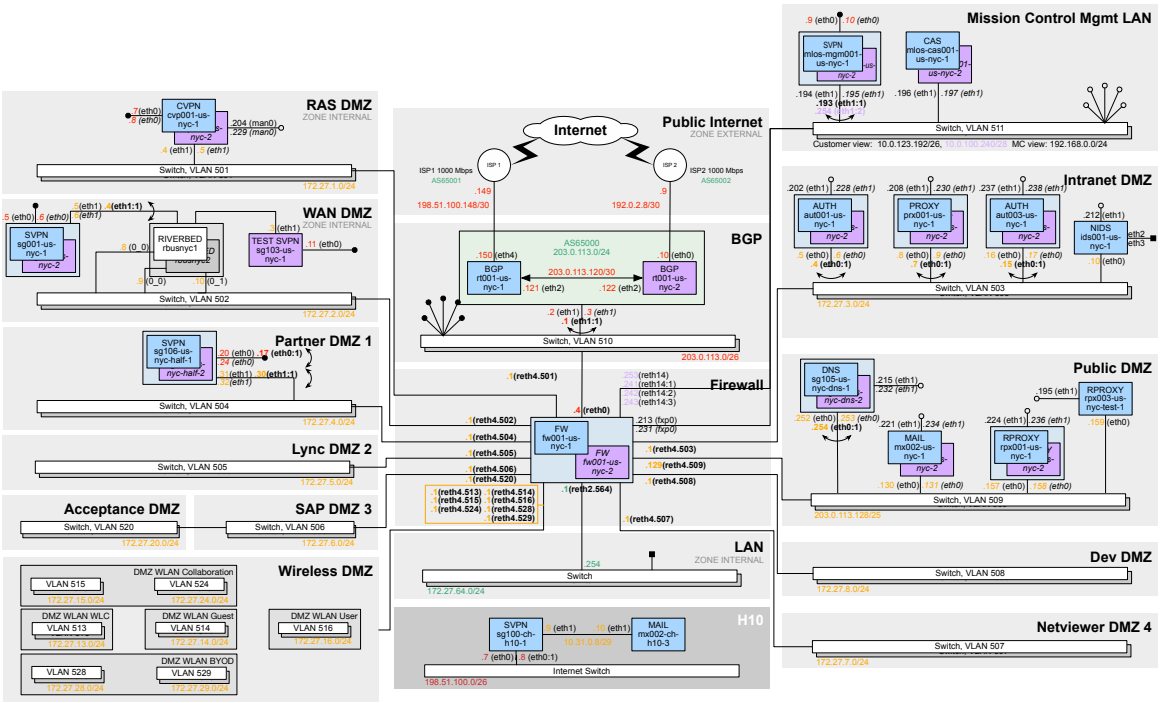


Figure 4.1: Generic network topology composed of a management LAN and management hosts on the upper-right corner, several DMZs, a LAN and specific devices such as firewalls or Border Gateway Protocol (BGP) peers.

The sensor hosts the data is gathered from have direct connection to the Internet. Most of these sensors consist of Management hosts (see Figure 5.5), they provide access to internal machines such as Web or Mail servers that would be located in DMZs.

4.1.1.2 Diversity of the Sensors

To have an accurate an unbiased overview of the IBN, the set of sensors that are used has to be heterogeneous to be representative. Besides covering every continent (see Figure 4.2), the sensors present a broad spectrum of connections such as multi-homed multi-gigabit connections with Tier-1 providers, digital subscriber line (DSL), local mini-DSL, Cable and Satellite providers, and BGP peers. The latter are responsible for exchanging routing and reachability information among Autonomous Systems (ASs). Various Internet Service Providers (ISPs) are also represented depending on where the host is deployed. From an attacker's point of view, the Internet connection type can make a difference since every ISP uses a particular connection type. ISPs often provide a single Internet connection type and the modem that is supplied is compliant with the connection type e.g. DSL or Cable. If a vulnerability is discovered on a specific type of modems, hackers will put more effort on the customer IP address space of ISPs using this modem thanks to publicly available allocation databases such as RIPE[1] or ICANN[2]. Consequently, a wider range of behavior classes can be cov-

[1] `https://apps.db.ripe.net/`
[2] `https://www.icann.org/`

ered. The locations of IP addresses of the sensors inside the /19 subnet (see Figure 4.3) owned by Open Systems AG are not heavily clustered, thus, have more chances in monitoring traffic destined to random IP addresses.

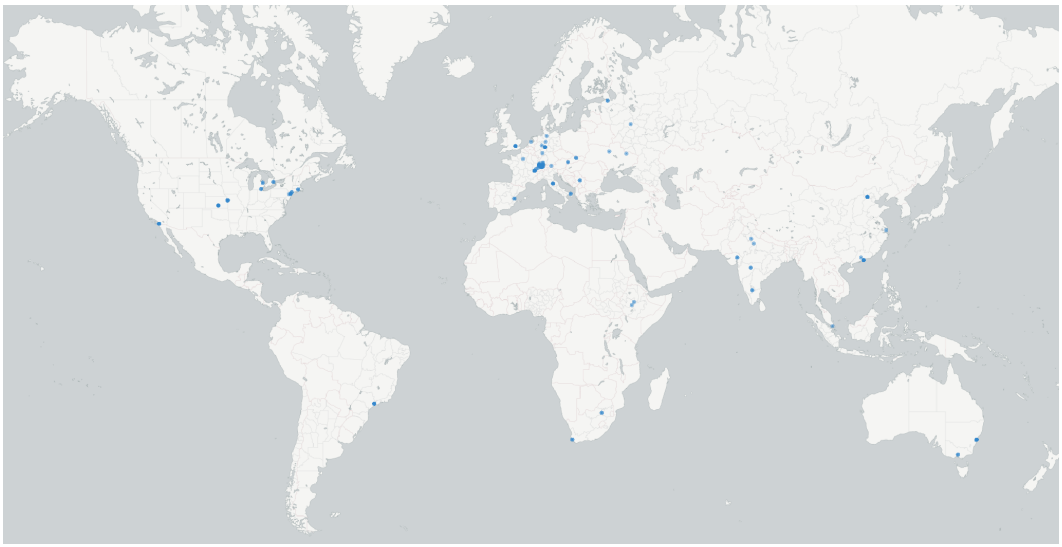


Figure 4.2: Geolocation of all hosts that are part of the IBN sensor network

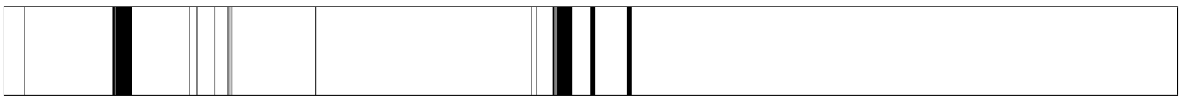


Figure 4.3: Spectrum representing the position of sensors in Open Systems's /19 subnet

4.1.1.3 `netfilter`

`netfilter`[3], which stands for Network Filter, is a flexible packet manipulation framework built into the Linux kernel. Together with `iptables`, they produce a modular packet selection system that enables the implementation of firewalls, Network Address Translation (NAT) and Port Address Translation (PAT) among others [49].

4.1.1.4 `iptables`

`iptables` is a user space command line program used to configure the Linux packet filtering ruleset and is commonly associated with `netfilter` [49] (see Section 4.1.1.3). The main features of `iptables` are: (1) listing the contents of the packet filter ruleset, and (2) adding/removing/modifying rules in the ruleset. The ruleset, which is called the *filter table*, is split into three built-in categories called *chains* (see Figure 4.5):

INPUT These rules are applied to filter packets that are destined to the machine.

FORWARD This chain handles packets where the machine is neither the destination nor the source.

OUTPUT These rules deal with packets that originate from the machine.

[3] `https://www.netfilter.org/`

According to their source and destination, packets traverse a specific chain. Packet header values are compared against the filter of each rule until one rule matches. This procedure is called *first-match* rule processing order; therefore, the order in which the rules are inserted matters. Two types of *actions* can be performed, either a *non-terminating action* or a *terminating action*. The action that has to be fulfilled is determined by the *target* part of a rule [84]. If a packet matches a rule that leads to a terminating action, every subsequent rule that would match this specific packet is ignored. The four relevant actions, in the scope of this work, are the listed below, where terminating actions are in bold:

ACCEPT The packet is accepted by the firewall filter and further processing of the packet can proceed.

DROP The packet is discarded without any response.

REJECT The packet is discarded and an error response is sent back.

NFLOG The packets that match have their header logged.

```
iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
```

(a) Accept all outgoing TCP packets for port 80

```
iptables -p INPUT DROP
```

(b) Drops all packets that reach the rule, i.e. implicit deny

```
iptables -A FORWARD -j NFLOG --nflog-group 32 --nflog-prefix "fw: log forward"
```

(c) Logs all the traffic of the FORWARD chain

Figure 4.4: Sample of `iptables` commands to create firewall rules

Figure 4.4 shows core commands used in the scope of this work. The command in Figure 4.4c allows to log all packet headers that match its filter, which, in this specific case, is every packet because the filter part of the rule is empty. NetFilter Log (NFLOG) is the logging tool used to document denied traffic. Logging of incoming traffic is done by adding an NFLOG target line before terminating actions that filter traffic on external interfaces only, on every chain of every host that is used as a sensor. Consequently, every packet that is implicitly dropped is logged. The second part of the command specifies how the logging works. The group number must match a corresponding group in `ulogd`[4]. `ulogd` is a user space logging daemon for `netfilter/iptables` related logging[4]. Several other parameters for the NFLOG [84] target such as `nflog-prefix` are used to provide more information, for example, to distinguish from which rule the packet was logged.

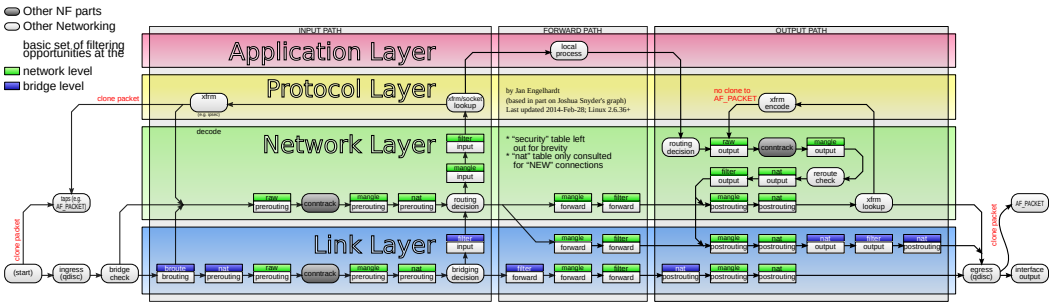


Figure 4.5: `netfilter` packet flow diagram showing the flow of packets through the different chains and tables. (Jan Engelhardt, 2014)

[4] `https://www.netfilter.org/projects/ulogd/`

4.1.1.5 `syslog`

The `syslog` protocol [34] is a standard for message logging and is used to convey event notification messages. It is intended to separate message content from message transport and allows extensibility for each layer. `syslog-ng`[5], an open source implementation of `syslog` can be used for creating centralized logging solutions. This solution is used to forward firewall logs from sensors, which are in this case *client* hosts, to a unique host that centralizes the logs. A *server* host, the aggregation host, is also setup to collect the logs. Logs are sent in nearly real-time to the aggregation host on port UDP/514.

4.1.2 Comparison with Existing Methods and Datasets

In the following subsections, the traffic trace dataset is compared with data used in previous research work (see Chapter 3). The origin of traffic data and the traffic capture level are both discussed.

4.1.2.1 Traffic Data Origin

Traffic data collected for this work resembles traffic gathered by monitoring darknets [7, 20, 21, 24, 72, 97] (see Figure 3.1). However, in the scope of this work, the monitored IP addresses are bound to real machines [51], rather than a part of a network telescope where IP addresses do not belong to a particular host. Most of the hosts deployed by Open Systems AG, in theory, are not reachable by an unauthorized machine. Taking a subset of 176 machines out of 4000 for a preliminary analysis looks also similar to Grey-IPs because these IPs are sparsely distributed among the whole address range [47, 48]. As a consequence, the sensor network gathering traffic can more likely observe malware with complex behavior, which is usually not caught by network telescopes. It is important to emphasize that traffic traces used in other research work almost always come from Internet backbones, central ISP points or big network entities such as campus networks where bidirectional flows can be observed [35, 47, 48, 73, 87, 98], whereas unidirectional traffic is analyzed in this work.

4.1.2.2 Capture Level

While previous research work does an application level analysis using full IPv4 packet capture [7, 72, 97], this work focuses on `netflow` trace analysis [20, 21, 35, 47, 48, 87, 73, 98] to scale with the traffic volume. As all involved hosts have a limited storage space, only packets coming from the Internet and not from the customer's network are recorded. Honeypots have been setup in [72, 97] to get a deeper understanding of cyber threats. Considering the real-time application of this work, deep packet analysis or active monitoring with honeypots does not apply here.

[5] `https://syslog-ng.org/`

4.1.2.3 Summary

Table 4.1 shows different datasets capture methodologies encountered in related work. Previous research work is categorized according to (1) the source of traffic data, (2) the packet capture level, and (3) the monitoring type, that states whether responses have been sent to IBN or not. Highlighted cells represent the methodologies used in the scope of this work.

Source	**Darknet**	Live Network
	[7, 20, 21, 24, 72, 97]	[35, 47, 48, 73, 87, 98]
Level	Full Packet Capture	**`netflow` Data**
	[7, 72, 97]	[20, 21, 35, 47, 48, 87, 73, 98]
Monitoring	Active	**Passive**
	[72, 97]	[7, 20, 21, 24, 35, 47, 48, 72, 73, 87, 97, 98]

Table 4.1: Summary comparison table showing the different traffic capture methodologies used in related work.

4.1.3 Data Overview

Firewall logs are stored in a text file and consist of traces of every denied IPv4 packet. An example of such a log line can be seen in Listing 4.1. The fields used for analysis are labeled. These different parameters are subject to change and may or may not have a value depending on the protocol used, e.g. source and destination ports are only part of Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) packets. Since traffic is denied and is not forwarded, the `OUT` interface is always empty. Listing 4.1 shows a denied `SYN` flagged TCP packet coming from `37.34.183.156` and destined to `46.140.65.139` on port 23, i.e. `telnet`. The packet was denied by the host `SENSOR_HOST_NAME` on October 4th around midnight (UTC+1). Its size without MAC header is 44 bytes and it entered the host on interface `eth0`.

```
<86>Oct 4 00:30:27 SENSOR_HOST_NAME ulogd[737]: fw: CILOG IN= eth0 OUT= MAC=00:10:f3:2d
    :35:26:e4:48:c7:74:03:5b:08:00 SRC=37.34.183.165 DST=46.140.65.139 LEN= 44
    TOS=00 PREC=0x00 TTL=50 ID=64497 PROTO= TCP SPT= 9597 DPT= 23 SEQ=780943755 ACK
    =0 WINDOW=46263 SYN URGP=0 MARK=0
```

Listing 4.1: A denied TCP packet with the `SYN` flag set

Traffic from 176 hosts has been chosen to simulate today's Internet Background Noise and analyze its composition. The main reasons to choose a subset of the hosts are to (1) demonstrate the feasibility of the extraction scheme presented in Chapter 6, (2) have less data to work on, therefore making the analysis faster, and (3) validate the global overview of IBN the traffic data gives. The data is organized by year, month and day on the aggregation host (see Figure 4.6). Data is parsed and aggregated before further analysis.

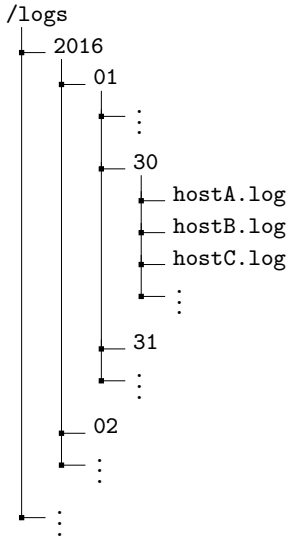


Figure 4.6: Directory structure on aggregation host. Log files are organized by year, month and day. Each file contains the packets which have been dropped by a host, on a specific day.

4.1.4 Data Parsing

To structure the data so that it is easier to manipulate and analyze, the raw log files are processed by a Python script that extract interest relevant fields with a regular expression. The extracted values are exported into Comma Separated Values files (`csv`). The following fields are extracted from the logs. As in the `netfilter` log files, every line represents the IP header of a dropped packet.

- Date and Time (`datetime`), Local Time in Zurich, Switzerland
- Host (`host`)
- Source (`src`)
- Destination (`dst`)
- Length (`len`)
- Protocol (`proto`)
- Source Port (`spt`)
- Destination Port (`dpt`)
- ICMP Type [11] (`type`)
- ICMP Code [11] (`code`)
- URG Flag (`urg`)
- ACK Flag (`ack`)
- PSH Flag (`psh`)
- RST Flag (`rst`)
- SYN Flag (`syn`)
- FIN Flag (`ack`)

The `host` field captures the name of the sensor host that dropped the packet. `len` represents the length of the IPv4 packet, without the link layer header, in bytes. Source and destination port values only apply to TCP and UDP packets. For protocols that do not make use of ports, the default value is -1. The same reasoning is applied for Internet Control Message Protocol (ICMP) Code and Type for non-ICMP packets. Regarding the TCP flags, their value is set to 1 if the flag is set in the packet header and 0 if not. A fortiori, all flag columns are set to zero for non-TCP packets.

4.2 Analysis Tools

Various software tools and additional Python libraries are used to conduct the analysis. They are presented in this section.

4.2.1 Python

Python [96] (version 3.6) was chosen as the data analysis scripting language because of its simplicity. It provides a lighter alternative to MATLAB[6] and R[7] with powerful data analytic libraries available (see Section 4.2.2).

4.2.2 Libraries

As stated in Section 4.2.1 powerful and easy-to-use analytic libraries are available. The additional libraries that are used are presented in the following subsections.

4.2.2.1 NumPy

NumPy [95] is the most commonly used and foundational package for scientific computations. It provides, among others, a powerful N-dimensional array (`ndarray`) object implementation. For numerical data, NumPy's arrays are much more efficient in storage and manipulation than other built-in Python data structures.

4.2.2.2 Pandas

Pandas [63, 64] is an open source library providing high performance, rich and easy-to-use data structure such as `DataFrame` and `Series` objects. Built on top of the NumPy (see Section 4.2.2.1) package, it provides the flexible data manipulations capabilities of relational databases, e.g. Structured Query Language (SQL).

4.2.2.3 Matplotlib and Seaborn

Matplotlib [42] is a Python 2D plotting library. Written in pure Python it makes heavy use of the NumPy library. It emulates MATLAB[6] graphics commands and overcomes it when applications grow in term of complexity, benefiting from Python's flexibility. Built on top of Matplotlib, Seaborn provides more attractive statistical graphics. Most graphs seen in this report are generated using these libraries.

4.2.2.4 PyFIM

PyFIM [10] is a Python library that provides a collection of data mining algorithm including some frequent itemsets mining implementations. An overview of these algorithms can be found in [9]. The implementation of the FP-Growth algorithm [36] is used for mining frequent itemsets.

4.2.2.5 Scikit-Learn

Scikit-Learn [74] provides several Machine Learning implementation algorithms including unsupervised and supervised learning methods, data mining and data analysis. This library is mainly used during the data exploration process of this work.

[6] `https://mathworks.com/products/matlab/`
[7] `https://www.r-project.org/`

5

Traffic Analysis

This chapter presents an overview of results that have been extracted from the past year's Internet Background Noise (IBN) collected at the 176 sensor hosts. The analysis is conducted on the traffic data collected between the 24th of February 2016 and the 31st of January 2017. The combined size of `bzip`[1] compressed traffic traces files is approximately 4.5 GB. A total of 286'705'280 Internet Protocol (IP)v4 packet headers coming from 19'051'653 different source IP addresses sent to 594 different destination IP addresses have been collected during the past year. This chapter starts with describing the composition of the traffic. Then, a temporal analysis of observed IBN is done through different visualizations on packets volume and protocol portion in the traffic. Finally, several case studies are presented by outlining frequent traffic patterns.

5.1 Traffic Composition

The composition of the collected IBN traffic is described in the following subsections. A monthly breakdown by protocol is described to get the proportion of the total traffic volume by transport layer protocol. Encountered Transmission Control Protocol (TCP) flags and Internet Control Message Protocol (ICMP) type and code combinations are then outlined. Eventually, recurrent targeted ports are summarized.

5.1.1 Traffic Breakdown by Protocol

Figure 5.1 and Table 5.1 show overall traffic month-based rate received by the sensor network. It can be noticed that TCP packets account for more than 60% of the traffic volume almost every month. The increase TCP traffic over the past year might be related to the spreading Internet of Things (IoT) botnets (see Section 5.3.1). User Datagram Protocol (UDP) and ICMP share the rest with variable portions over the different months. Other less represented protocols such as Generic Routing Encapsulation (GRE), Encapsulating Security Payload (ESP), Internet Group Management Protocol (IGMP), Stream Control Transmission Protocol (SCTP), Resource Reservation Protocol (RSVP) and IPv6 encapsulation are encountered over time. It may be important to remark that `nmap` [59], a network discovery software implements SCTP and IGMP scan schemes in its toolbox. In total 3036 IGMP packets have been denied. The majority of the traffic was received between February 24th and February 25th from six misconfigured devices sending packets to 224.0.0.1, the Multicast address. Ten other IGMP packets coming from three different sources and destined to six destinations are recorded for the months of May, December and January 2016. IPv6 encapsulated traffic mainly came from 6to4 relays, i.e. sources in 192.88.99.0/24, which transfer traffic from presumably mis-

[1] `http://www.bzip.org/`

configured devices. However, twelve packets coming from the same source in Russia and destined to different destinations have been received on October 11th, 2016. This source has been classified as a scanner by the publicly available database IBM X-Force Exchange [44]. ESP and GRE traffic does not appear to be malicious and is probably due to misconfigurations. RSVP traffic is mostly generated by a misconfigured device in Mexico which sends a heartbeat every day.

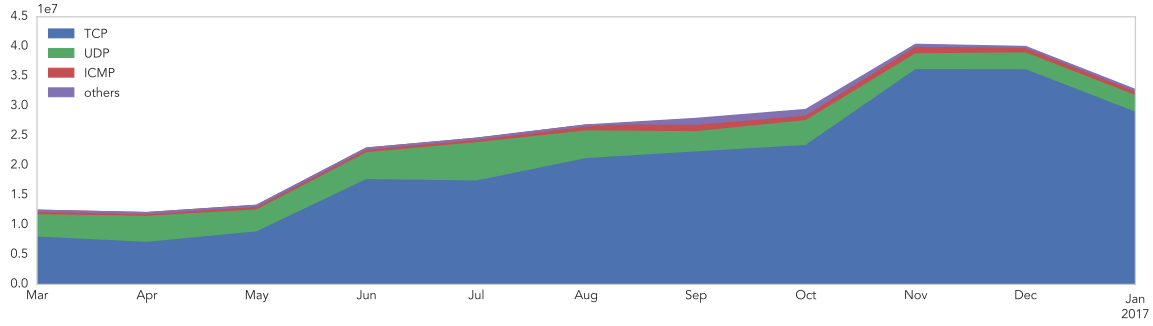


Figure 5.1: Monthly observed IBN traffic volume broken down by most represented protocols.

Month	Packets	TCP	UDP	ICMP
03-16	12'474'490	63.46	30.26	6.26
04-16	12'040'363	58.14	36.46	5.37
05-16	13'238'985	66.07	27.92	5.96
06-16	23'120'217	76.76	19.76	3.46
07-16	24'537'804	70.52	26.31	3.15
08-16	26'751'667	78.76	17.41	3.81
09-16	28'168'078	79.70	12.22	3.81
10-16	30'323'533	79.31	14.28	2.51
11-16	41'252'960	89.38	6.62	2.67
12-16	40'049'573	90.15	7.11	2.11
01-17	33'040'302	88.26	8.57	1.80

Table 5.1: Monthly traffic breakdown by protocol in percent

5.1.1.1 TCP Flags

Diverse combinations of TCP flags occurred over the whole traffic trace (see Table 5.2) and many of these combinations can be used to perform port scanning [52]. The most frequent flag combination is indisputably the one with the sole `SYN` flag set. This scan method is quick, unobtrusive and stealthy since the TCP handshake is not completed. Furthermore, it allows reliable differentiation between open, closed and filtered ports. When only the `ACK` flag is set, an attacker can determine if a port is filtered or not by a firewall and if it is stateful or not. TCP `RST` flagged packets may be `RST` attacks trying to terminate a valid connection between two hosts. Other less frequent combinations could be either custom scans or misconfigured devices sending bogus traffic. According to [67], the encountered combinations are not part of typical denial of service (DoS) backscatter traffic (see Table 5.2 and Section 2.3.4.1).

URG	ACK	PSH	SYN	FIN	RST	Sources	Packets	Hypothesis
			■			95.80	99.81	`SYN` Scan [52]
■						3.43	0.11	`ACK` Scan [52]
■	■					0.66	0.05	Misconfiguration
	■	■				0.02	0.03	Custom
■				■		0.52	0.00	`FIN` Scan [52]
					■	0.48	0.00	`RST` Attack [52]
■					■	0.01	0.00	Misconfiguration
■	■			■		0.00	0.00	Misconfiguration
■			■			0.00	0.00	Misconfiguration
■	■					0.00	0.00	Misconfiguration
■					■	-	-	DoS Backscatter [67]
■			■			-	-	DoS Backscatter [67]
					■	-	-	DoS Backscatter [67]
						-	-	Null Scan [52]
				■		-	-	`FIN` Scan [52]
■	■				■	-	-	X-mas Scan [52]

Table 5.2: Encountered flag combinations with monthly average percentage of sources and packets among TCP traffic. Frequent combinations which can be encountered in IBN but not in the analyzed traffic are also shown with dashes instead of values.

Table 5.2 also shows frequent TCP flag combinations that can be part of IBN but that are not part of the analyzed traffic data. TCP flags can help identify sources under a DoS attack [67]. Null, `FIN` and X-mas scans exploit a loophole in the TCP protocol to find out if a port is open or not [52]. These scan techniques are often employed for bypassing firewalls and sometimes evading Intrusion Detection System (IDS).

5.1.1.2 ICMP Types and Codes

ICMP packets represent a nonnegligible amount of the traffic. Each packet carries a type and a code value [11]. Table 5.3 shows the combinations of values that have been encountered during the analysis, in descending order by number of packets, with a brief supposition of the origin of this traffic.

Type	Code	Packets	Comments
8	0	9197111	ECHO Request (see Section 2.3.3).
13	0	25759	Timestamp Request, can be used to ping hosts.
8	9	501	Undefined/Illegal, may be a misconfigured device
17	0	377	Address Mask Request (Deprecated), can be used for fingerprinting [58].
15	0	288	Information Request (Deprecated), can be used for fingerprinting [90].
3	3	282	Destination Unreachable, can be Backscatter traffic.
8	123	228	Undefined/Illegal, may be a misconfigured device
8	30	8	Undefined/Illegal, may be a misconfigured device
8	8	4	Undefined/Illegal, may be a misconfigured device

Table 5.3: Encountered ICMP Type and Code combinations with the number of packets across the whole traffic data.

5.1.1.3 Size of Packets

Multiple packet sizes have been encountered during the monitoring phase. Figure 5.2 shows the byte length distribution of received IPv4 packets without the link layer header. The distribution is positively, right, skewed, showing that more than 80% of the packets are less than 60 bytes. Both IPv4 [77] and TCP [79] headers are between 20 and 60 bytes. UDP [76] and ICMP [78] headers are 8 bytes each. This confirms that the major part of the observed IBN is malicious scanning with little payload packets. The minimal packet size encountered is 21 bytes and one packet of 30'700 bytes is the maximal size observed. Four packets between 26'000 and 30'700 bytes came from the same source on January 4th, 2017 targeting port TCP/3433. This address was probably looking for Linux hosts which still had the TCP stack implementation vulnerability (see Section 5.3.4).

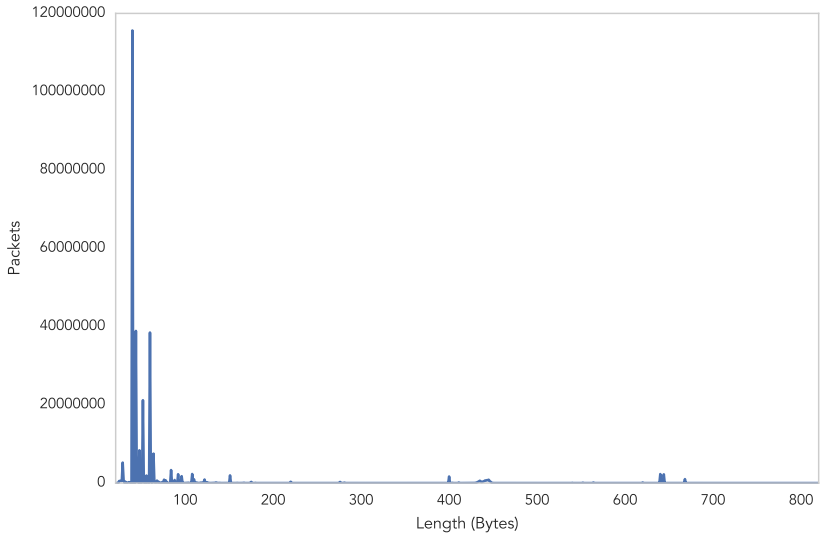


Figure 5.2: Packet length distribution for packets which are less than 800 bytes. Clear clusters are distinguishable under 100 bytes confirming the proportion of malicious traffic.

5.2 Temporal Analysis

This section studies temporal differences of different aspects of IBN. At first, the focus is on the evolution of targeted destination ports for both TCP and UDP trying to infer the cause of this traffic by matching known vulnerabilities encountered in the past months. Then, special attention is paid to traffic volume evolution in terms of number of packets and size of log files.

5.2.1 Details per Ports

The following subsections describe the top targeted TCP and UDP ports over time. The presented statistics and visualizations give an overview of the trending threats and vulnerabilities mainly encountered during the last year. Heatmaps of most targeted destination ports for both TCP and UDP across time are presented in Figures 5.9 and 5.10.

5.2.1.1 TCP Ports

Table 5.4 shows a month based evolution of the targeted TCP destination ports. Ports are sorted in descending order according to the number of unique sources that scanned the port. Indisputably, the top targeted TCP destination port is port 23, which is the standard port used by Telnet [62]. Devices running a Telnet service are preys of brute-force password guessing attacks [23] (see Section 5.3.1) along with Secure Shell (SSH) [50], on port TCP/22, Simple Mail Transfer Protocol (SMTP), on port TCP/25, or Virtual Network Computing (VNC) [80] servers.

TCP/445 is another predominant targeted destination port during the first half of the monitoring phase. TCP/445 and UDP/445 are used by Microsoft related applications such as Server Message Block (SMB), previously known as Common Internet File System (CIFS), or Active Directory and are prone to vulnerabilities [CVE-2016-3345, CVE-2016-3225, CVE-2016-0037, CVE-2016-3226]. Part of the Microsoft application suite, Remote Desktop Protocol (RDP), on port TCP/3389, has suffered exploitable vulnerabilities including privilege escalation, security bypass and information disclosure [CVE-2016-0036, CVE-2016-0019, CVE-2016-0190]. Microsoft Structured Query Language (SQL) Server (TCP/1433) was subject to six different vulnerabilities during 2016 alone [CVE-2016-7254, CVE-2016-7253, CVE-2016-7252, CVE-2016-7251, CVE-2016-7250, CVE-2016-7249] including privilege elevation and Cross-site Scripting (XSS). All aforementioned vulnerabilities are exploited by attackers which is why they appear in the top 10 scanned ports shown in Table 5.4.

Port scanning targeting TCP/3306, MySQL servers, which happens in February 2016, might be credential guessing attacks [70] because there is no record of any vulnerability affecting this application for this time period. Same reasoning applies to port TCP/6379, which is often bound to the Redis database application and might be linked to White hats protesting about the use of default credentials [86].

Usual Web application ports such as TCP/80, TCP/8080, and TCP/443 are also targeted. Such traffic might be scanners that look for Web Applications vulnerabilities (see Section 5.3.3) such as XSS, SQL Injections, Command Injection [88], or insecure server configurations.

Between June and August 2016, TCP/2222 was part of the top 10 targeted ports. Indeed, Rockwell, an application that is bounded to the port, has suffered many critical vulnerabilities during 2016 including buffer overflows that permit remote code injection [CVE-2016-0868, CVE-2016-2277, CVE-2016-5814], XSS [CVE-2016-2279], or SQL injection [CVE-2016-4522]. After August 2016, the focus went on other ports, thus TCP/2222 was not part of the top 10 until January 2017 where a burst of new sources targeted this port (see Figure 5.3). Further investigations have been done and it appeared that all destination hosts were targeted evenly by a high number of sources. This may reflect a global IPv4 horizontal scan since no critical vulnerability affecting Rockwell services was published at this time. These scans may also be SSH password guessing attacks. Indeed TCP/2222 is sometimes used as an alternative to TCP/22.

Finally, TCP/23, TCP/2323, TCP/7457, TCP/5555, TCP/23231, TCP/6789, TCP/37777, and TCP/5358 are all target ports of the cracking IoT botnet Mirai and its derivatives (see Section 5.3.1).

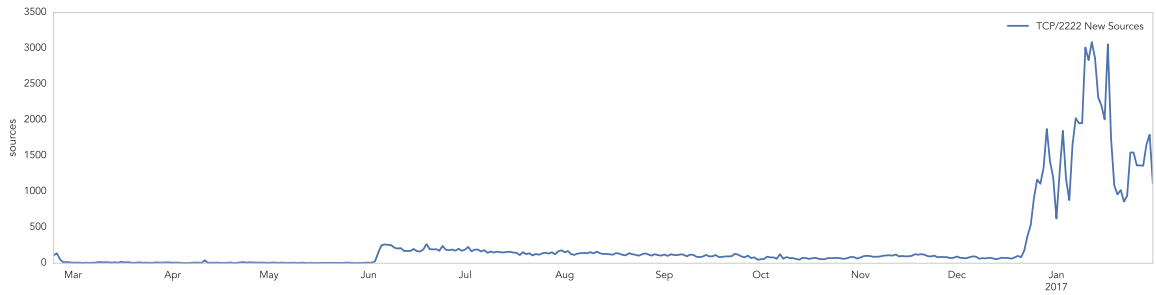


Figure 5.3: Daily observed new sources targeting TCP/2222

02-16	03-16	04-16	05-16	06-16	07-16	08-16	09-16	10-16	11-16	12-16	01-17
21	21	21	21	21	22	22	21	21	21	22	22
22	22	22	22	22	23	23	22	22	22	23	23
23	23	23	23	23	80	80	23	23	23	80	445
25	25	80	25	80	443	443	80	80	80	445	2222
80	80	443	80	445	445	445	443	443	445	2323	2323
443	443	445	445	1433	1433	1433	445	445	2323	5358	5358
445	445	1433	1433	2222	2222	2222	1433	1433	3389	5555	6789
3306	1433	3389	3389	3389	3389	3389	2323	2323	5555	6789	7547
3389	3389	4028	4028	4028	4028	4028	3389	3389	7547	7547	23231
8080	4028	8080	6379	8080	8080	8080	8080	8080	8080	23231	37777

Table 5.4: Top 10 TCP destination ports by number of unique sources month based evolution. Ports are sorted by port number and colors indicate top three ports triplets. Red indicates the top port, green the second top port and blue the third top port by unique sources.

5.2.1.2 UDP Ports

The top 10 UDP targeted destination ports evolution across months is shown in Table 5.5. Only important trends and vulnerabilities are outlined given the spectrum of port numbers. A clear trend between February and August 2016 for the top two ports is noticeable, namely UDP/53413 and UDP/137. Port 53413 [93] is a long-time known backdoor in Netis routers first described in 2014. This brand of routers is mostly present in China but they can be found in other parts of the world. Any payload sent to UDP/53413 of such device is automatically executed. The traffic is probably generated by a worm which is infecting vulnerable targets to turn them into scanners. On Shadowserver foundation's initiative, a project [30] was launched to identify vulnerable systems and report them to their owner.

UDP/137 has also been a predominant target port in the past year (2016), associated with a vulnerability affecting all Microsoft Windows versions from Windows 95 (1995) to Windows 10 (2015) has been found by Yu [100], a Chinese researcher. Nicknamed *BadTunnel* [100] by its finder, this vulnerability allows an attacker to get access to network traffic without being on the victim's network, and bypass firewall as well as Network Address Translation (NAT) devices.

An increasing number of sources have been targeting UDP/1900 at the end of 2016. Simple Service Discovery Protocol (SSDP) has become more popular than Network Time Protocol (NTP) [37] (UDP/123) for Domain Name Service (DNS) amplification attacks. SSDP is attractive for distributed denial of service (DDoS) attacks because it allows IP spoofing. By eliciting a server response directed to the victim which is disproportionate to the original packet. Therefore, it is most probably traffic generated by botnets looking for machines with this port open in preparation for future DDoS attacks.

Traffic targeted to UDP/2222 during September 2016 could be blamed on scanners trying to exploit a Cisco IOS Software vulnerability [CVE-2016-6391] that was affecting the Common Industrial Protocol (CIP). This vulnerability could allow an unauthenticated, remote attacker to create a DoS condition.

Port UDP/27015, often attributed to the Half-Life Server application, has been heavily targeted in November 2016. Since no other service bounded to this port has been found, attackers may be trying to exploit a vulnerability [CVE-2007-5713] dated of 2007 which allows to execute arbitrary code or cause DoS without authentication. UDP/3544 is used by Teredo, an IPv4 to IPv6 transition technology that can be prone to exploits that can provide a way to bypass firewall and perimeter defenses for attackers [39]. An exploit on specific Asus routers allows attackers to bypass authentication and run arbitrary commands by sending crafted packets to port UDP/9999 [CVE-2014-9583].

Even though the vulnerability has been fixed quickly, scanners keep trying to find out-of-date exploitable routers. Packets destined to port UDP/123, i.e. to NTP, are most likely trying to exploit known vulnerabilities as this protocol is often affected by bugs that can cause DoS [1]. Traffic directed to port 53 might be scanners looking for open resolvers [31] foreseeing DNS amplification attacks. Finally, UDP/161, which is assigned to Simple Network Management Protocol (SNMP), is often used for reconnaissance purposes [81] because SNMP is frequently installed on systems by default.

02-16	03-16	04-16	05-16	06-16	07-16	08-16	09-16	10-16	11-16	12-16	01-17
53	137	137	137	53	53	28	53	53	53	53	137
123	8303	8306	175	123	123	53	123	137	123	137	1900
137	12036	13998	176	137	137	123	137	176	137	1900	3544
161	22815	14383	3544	3544	3544	137	3544	8306	1900	3544	8302
5060	40098	30247	14382	5060	5060	3544	5060	12036	3544	5060	14383
33434	40099	53413	14383	9999	9999	5060	14382	12037	5060	12037	25792
37470	53413	55238	36026	43130	22559	9999	22222	22816	8302	14382	25793
40099	55238	58182	40099	50321	29479	40708	25793	25793	12036	16854	36027
40102	63577	63577	53413	53138	33434	53413	49631	27015	27015	49631	40102
53413	64991	64991	55238	53413	53413	64536	53413	28032	40098	55238	53413

Table 5.5: Top 10 UDP destination ports by number of unique sources month based evolution. Ports are sorted by port number and colors indicate top three ports triplets. Red indicates the top port, green the second top port and blue the third top port by unique sources.

5.2.2 Traffic Volume Evolution

Figures 5.1 and 5.7 present TCP traffic increase that is caused by the emergence of IoT botnets (see Section 5.3.1). The following subsections describe and explain this increase through different aspects such as the intrinsic growth of the log files and the rise in the volume of traffic (see Figure 5.7) despite the fact that the number of sensors remain stable during the monitoring phase (see Figure 5.8).

As the traffic volume increased, the size of `bz2` compressed log files also grew. Indeed, the growing of compressed log files is caused by log files containing more lines because of more denied IBN traffic, thus confirming the growth of IBN. Figure 5.4 shows the evolution log file sizes daily by host type where it can be noticed that, even if the majority of the hosts are Management hosts (see Figure 5.5), dedicated Firewall devices contribute more in terms of traffic volume than any other type of hosts. There are two explanations: (1) firewall hosts are covering more devices, consequently, more destination IP addresses (see Figure 5.6), and (2) these destination IP addresses are more likely to be publicly available resources such as Web servers, thus more exposed.

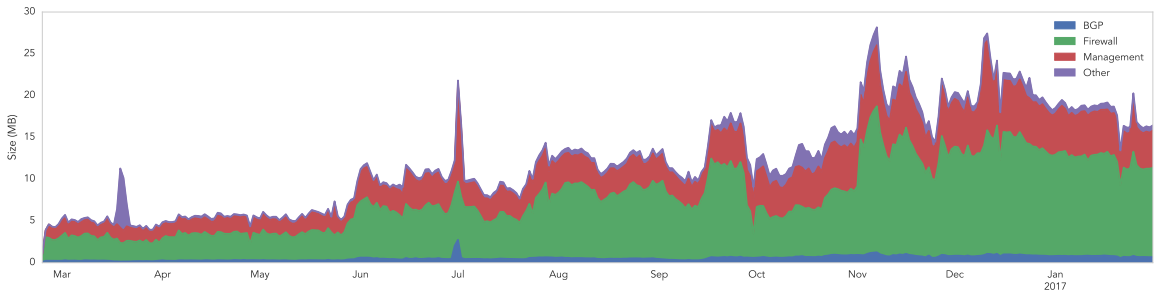


Figure 5.4: Log files size summation per day according to host types. Border Gateway Protocol (BGP) routers, Firewall devices and Management hosts are the main host categories constituting the sensor network. The remaining hosts mostly consists of hosts which are part of Open Systems's Wide Area Network (WAN) architecture such as Virtual Private Network (VPN) devices.

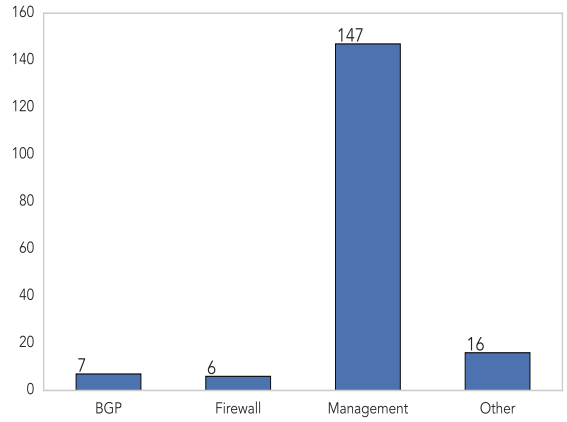


Figure 5.5: Distribution of the 176 sensor hosts by service type. The remaining hosts mostly consists of hosts which are part of Open Systems's WAN architecture such as VPN devices.

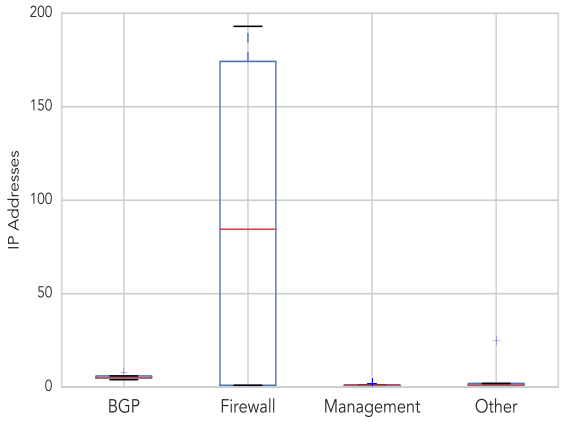


Figure 5.6: Distribution of the of number of covered IPs by host type. Box plots show extreme values, median and average number of IP addresses for each host type.

	Min	Median	Mean	Max	σ
BGP	4	5	5.6	8	1.52
Firewall	1	84.5	90.75	193	104.14
Management	1	1	1.14	2	7.16
Other	1	1	3.07	25	6.60

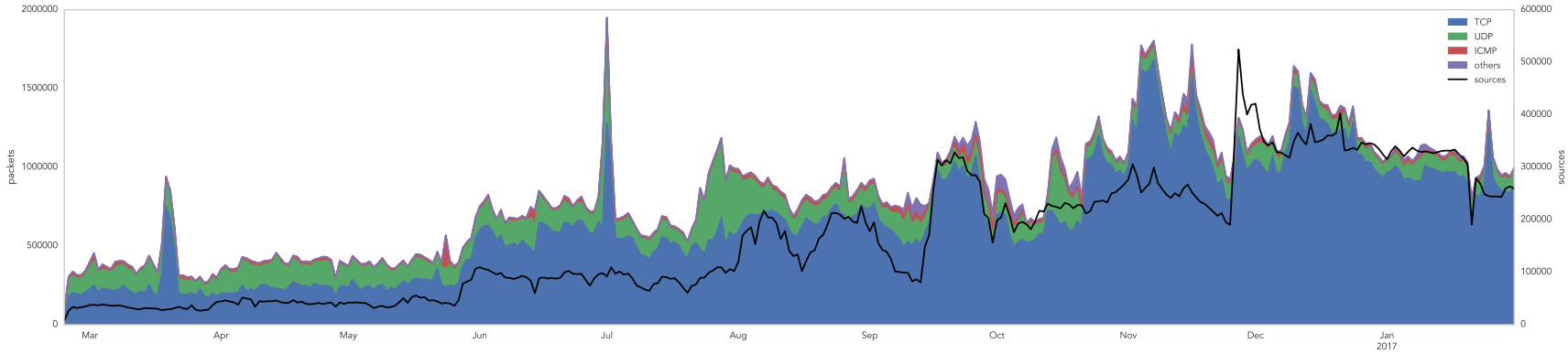


Figure 5.7: Daily observed IBN traffic volume broken down by protocol type

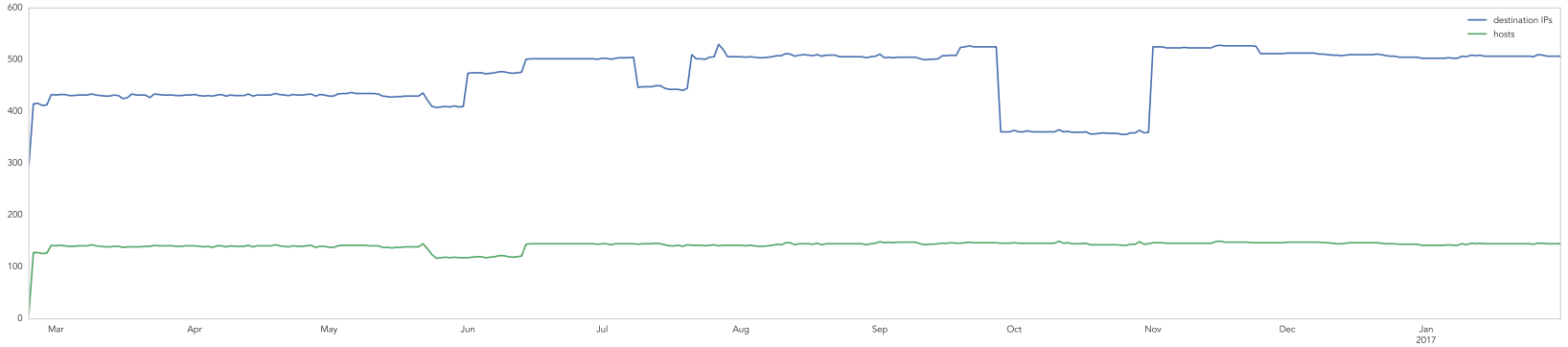


Figure 5.8: Daily targeted number of destinations IP addresses along with number of hosts. Monitoring had to be suspended on a firewall host for the whole month of October 2016. This host covers around 200 different IP addresses. Therefore, the number of monitored IP addresses dropped while the number of hosts was stable. Other variations result from similar scenarios.

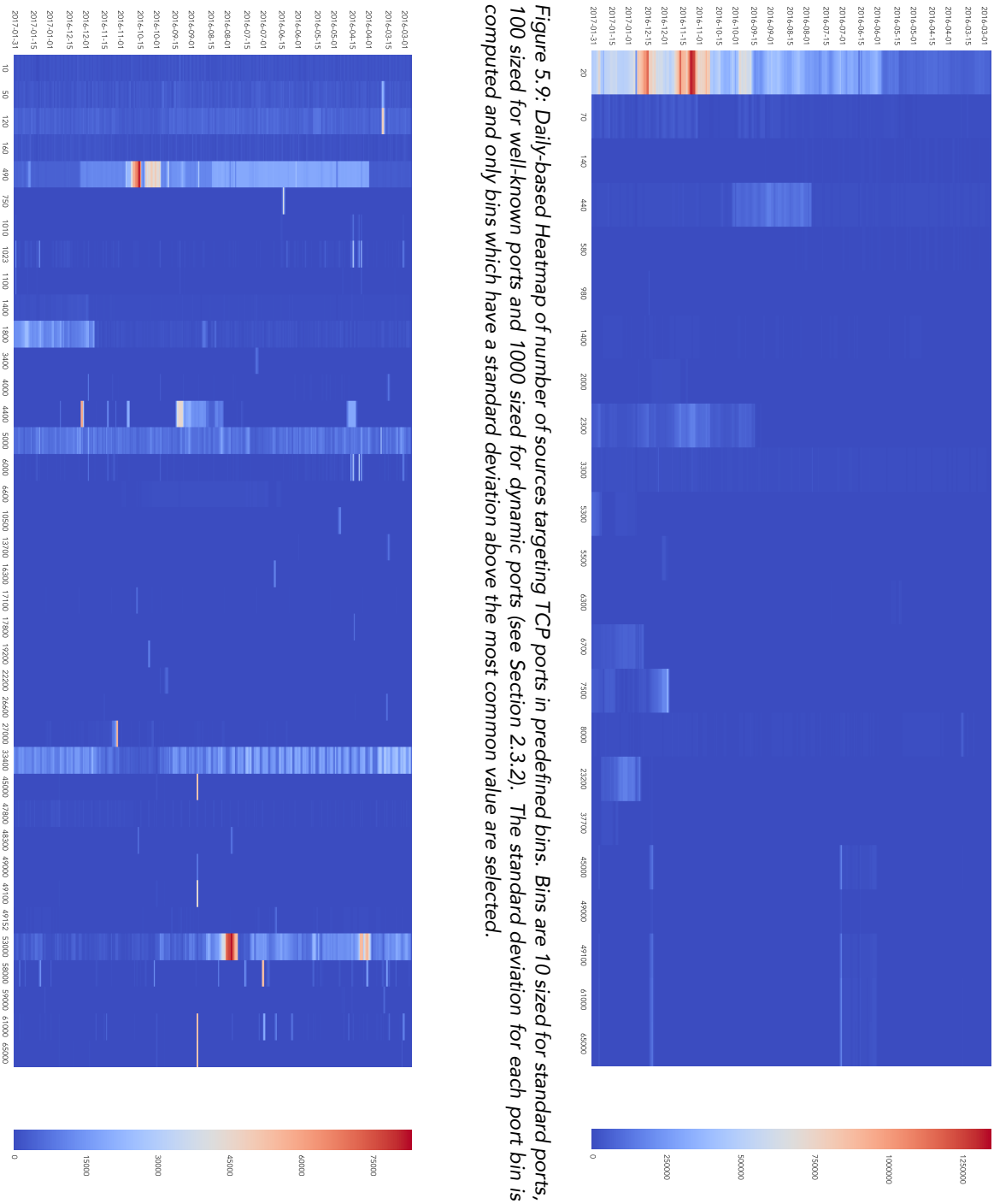


Figure 5.9: Daily-based Heatmap of number of sources targeting TCP ports in predefined bins. Bins are 10 sized for standard ports, 100 sized for well-known ports and 1000 sized for dynamic ports (see Section 2.3.2). The standard deviation for each port bin is computed and only bins which have a standard deviation above the most common value are selected.

Figure 5.10: Daily-based Heatmap of number of sources targeting UDP ports in predefined bins. Bins are 10 sized for standard ports, 100 sized for well-known ports and 1000 sized for dynamic ports (see Section 2.3.2). The standard deviation for each port bin is computed and only bins which have a standard deviation above the most common value are selected.

5.3 Case Studies

This section describes a selection of the past year's IBN behavior trends paying particular attention to Mirai, the cracking IoT botnet. As Voice over IP (VoIP) and Video Conference are widely used in corporate environments, Session Initiation Protocol (SIP) scanners behavior is also analyzed in more detail.

5.3.1 Mirai: An IoT Botnet

In September 2016 a new worm began to target Linux embedded IoT devices such as IP CCTV cameras and digital video recorders (DVRs). These devices often have Telnet or SSH configured with the default credentials, but are not actively using these services. The Mirai malware uses this information to infect the devices. As many other botnets, the core capability of the first version of the Mirai malware is to compromise new devices to expand the botnet. The main purpose of Mirai is to perform DDoS attacks (see Section 5.3.1.3). The worm is coded in two programming languages; the bot part is written in C and the Command and Control (C&C) server in Go [4].

5.3.1.1 General Evolution

By comparing Figure 5.7 and Figure 5.13, insight can be gained into the contribution of IoT botnets to IBN during the past year. Aside from the traffic volume growth, an increase is also noticeable in the number of different sources encountered (see Figures 5.7 and 5.13) and the proportion of sources which can be categorized into Mirai-like sources across time (see Figure 5.12). Between September and November 2016, Mirai infected hosts were contributing to most of the gathered IBN traffic with a ratio above 90%. The leakage of the source code on September 30th, 2016 [4] probably had a role in the increase of IBN, since everyone with little knowledge could run his own botnet. An increase in the number of sources targeting TCP/23 can be noticed right before the leak (see Figure 5.11), this might explain the publication of the source code. Publishing the code online ensures that the authors are not the only ones who can be found possessing it in case the authorities investigate [54].

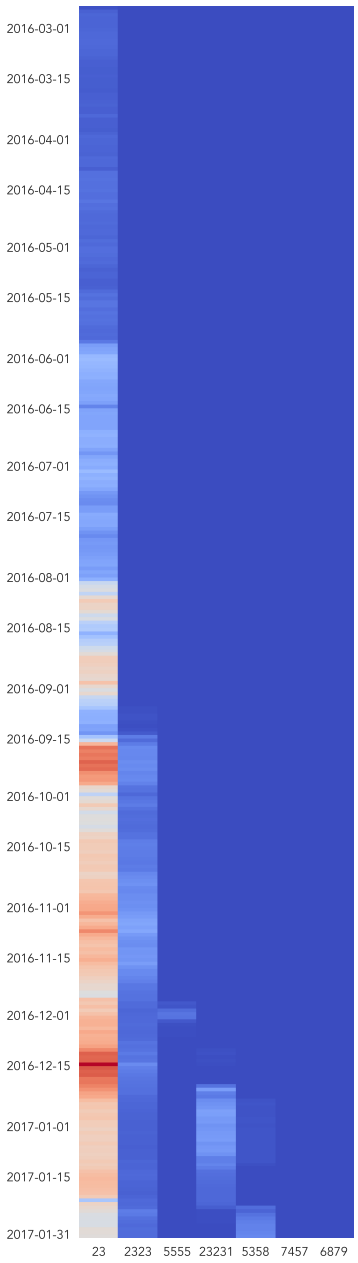


Figure 5.11: Waterfall graph representing the targeted ports of the Mirai botnet/worm and the number of distinctive sources targeting these ports per day.

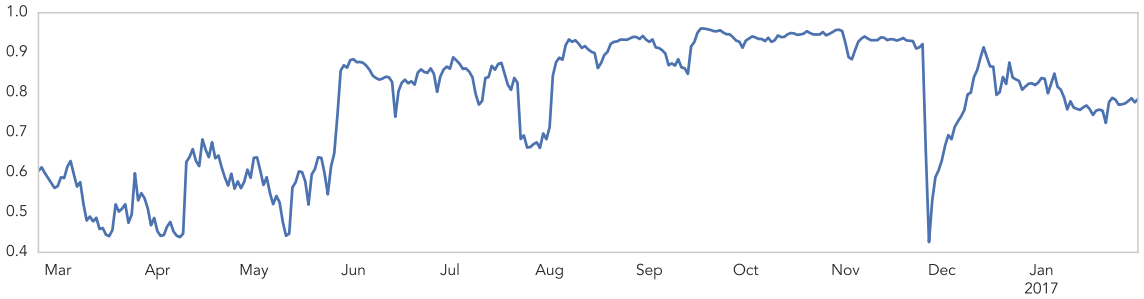


Figure 5.12: Daily proportion of Mirai-like classified sources over total number of sources. A source IP address is classified as Mirai-like if it scans at least one port which is known to be targeted by the botnet or its variations.

5.3.1.2 Target Ports

Figure 5.13 shows the observed traffic volume evolution of the botnet's destination ports target vector. The subset of traffic which is directed to publicly known destination ports targeted by Mirai is selected to show the volume evolution of the different ports. All sources encountered in this traffic are classified as Mirai-like sources and the number per-day of such sources is shown on Figure 5.13. Mirai-like botnets evolved taking advantage of new vulnerabilities (see Section 5.2.1.1) and targeting diverse destination ports (see Figure 5.11). The first targeted ports, as the original source code shows, are Telnet ports TCP/23 and the alternative TCP/2323 [23] every tenth probe (see Listing 5.1). The worm targets mainly IoT devices by performing password guessing attacks with the help of a default credential list [23] (see Listing 5.2). An interesting fact is that the worm avoids specific ranges of IP addresses [4]. This may expose concerns of drawing security community's attention [38].

```
if (i % 10 == 0)
{
        tcph->dest = htons(2323);
}
else
{
        tcph->dest = htons(23);
}
```

Listing 5.1: Mirai source code portion which chooses the destination port, one of every ten packet is destined to probe port TCP/2323 instead of TCP/23 [4].

```
// Set up passwords
add_auth_entry("\x50\x4D\x4D\x56", "\x5A\x41\x11\x17\x13\x13", 10);   // root   xc3511
add_auth_entry("\x50\x4D\x4D\x56", "\x54\x4B\x58\x5A\x54", 9);   // root   vizxv
...
add_auth_entry("\x56\x47\x41\x4A", "\x56\x47\x41\x4A", 1);   // tech   tech
```

Listing 5.2: Mirai source code portion which sets the credentials used during probes. There are 62 different user/password combinations in the original leaked source code [4].

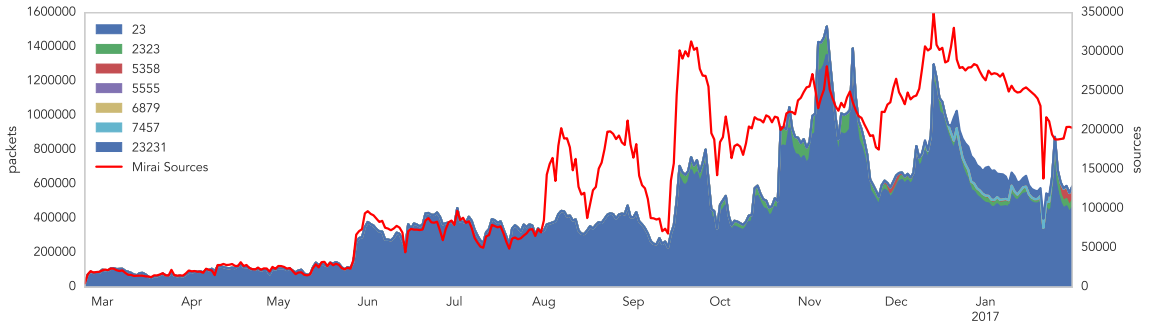


Figure 5.13: Mirai botnet classified traffic daily evolution with number of Mirai-like behaving sources. A source IP address is classified as Mirai-like if it scans at least one port which is known to be targeted by the botnet or its variations.

5.3.1.3 Attack Vector

IoT botnets caused several major DDoS attacks in the past year (2016). The original version of the Mirai malware implements several flooding functions (see Listing 5.3).

```
1   add_attack(ATK_VEC_UDP, (ATTACK_FUNC)attack_udp_generic);
2   add_attack(ATK_VEC_VSE, (ATTACK_FUNC)attack_udp_vse);
3   add_attack(ATK_VEC_DNS, (ATTACK_FUNC)attack_udp_dns);
4   add_attack(ATK_VEC_UDP_PLAIN, (ATTACK_FUNC)attack_udp_plain);
5
6   add_attack(ATK_VEC_SYN, (ATTACK_FUNC)attack_tcp_syn);
7   add_attack(ATK_VEC_ACK, (ATTACK_FUNC)attack_tcp_ack);
8   add_attack(ATK_VEC_STOMP, (ATTACK_FUNC)attack_tcp_stomp);
9
10  add_attack(ATK_VEC_GREIP, (ATTACK_FUNC)attack_gre_ip);
11  add_attack(ATK_VEC_GREETH, (ATTACK_FUNC)attack_gre_eth);
12
13  //add_attack(ATK_VEC_PROXY, (ATTACK_FUNC)attack_app_proxy);
14  add_attack(ATK_VEC_HTTP, (ATTACK_FUNC)attack_app_http);
```

Listing 5.3: Mirai source code portion which defines the different attack types [4].

Listing 5.3 shows attack methods provided by the malware. Common flooding functions, such as UDP, TCP `SYN`, TCP `ACK`, DNS, and Hypertext Transfer Protocol (HTTP) flooding can be found alongside other less known attacks, e.g. GRE or Valve Source Engine flood, on line 2, which is an amplification attack involving gaming servers. Compromised devices wait for commands coming for the C&C server.

On September 20th 2016, alongside with BASHLITE, another IoT, botnet, Mirai was involved in the record 620 Gbps DDoS attacks, on the KrebsOnSecurity [53] website and 1 Tbps on OVH, a French web hosting provider. October 21st 2016, DynDNS was under a DDoS attack [99] causing major websites such as CNN, BBC, Twitter, Netflix, Reddit and GitHub inaccessible. On November 3rd 2016, Liberia, the West African country, was offline because of a major DDoS attack perpetrated by Mirai on local Internet Service Providers (ISPs). The same happened to Deutsche Telekom which was disrupted causing problems for close to a million customers at the end of November 2016. The drop of the proportion of Mirai-like sources among IBN generators at the end of November 2016 that can bee see in Figure 5.12 might be related to this latter attack. Since Internet was down for many areas in Germany, it paradoxically prevented some scanning traffic. These attacks were highly distributed and involved millions of IP addresses which are associated with the Mirai botnet [69].

5.3.2 Session Initiation Protocol

Session Initiation Protocol (SIP) is an application-layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls [83], also known as Voice over IP (VoIP).

5.3.2.1 SIP Scanners

SIP scans on port UDP/5060 are also present in the analyzed traffic portion. Dainotti et al., in [21], studied a horizontal SIP scan performed by the Sality botnet, which targeted the entire IPv4 address space in February 2011 (see Section 3.2.3). These scanners target primarily VoIP-related infrastructures in two steps: (1) locate SIP servers, and (2) try to compromise them [21]. SIP scanners studied in [21] send a UDP packet to port 5060 carrying a SIP header which tries to register a random user account on a SIP server. Usually, SIP servers do not accept random account registration. Therefore, if the targeted host is a SIP server, the request will most likely fail and result in a *404 Not Found* response. At this point, the attacker has identified a SIP server which can be attacked by, for example, performing brute-force attempts to register a VoIP device [21]. In the traffic data 6966 different source IP addresses have been seen targeting UDP/5060. Indeed, during last year, several vulnerabilities in diverse implementations of the protocol have been discovered in Cisco devices such as memory leak vulnerability [CVE-2016-1350], or DoS vulnerabilities [CVE-2016-1338, CVE-2016-1350, CVE-2016-1400, CVE-2016-1466]. As shown by Figure 5.14, the daily packet volume as well as the number of sources remain stable throughout the time of the analysis. It is believed that most of these scanners are indeed malicious, however, some probes might originate from security companies or security foundations analyzing vulnerable hosts.

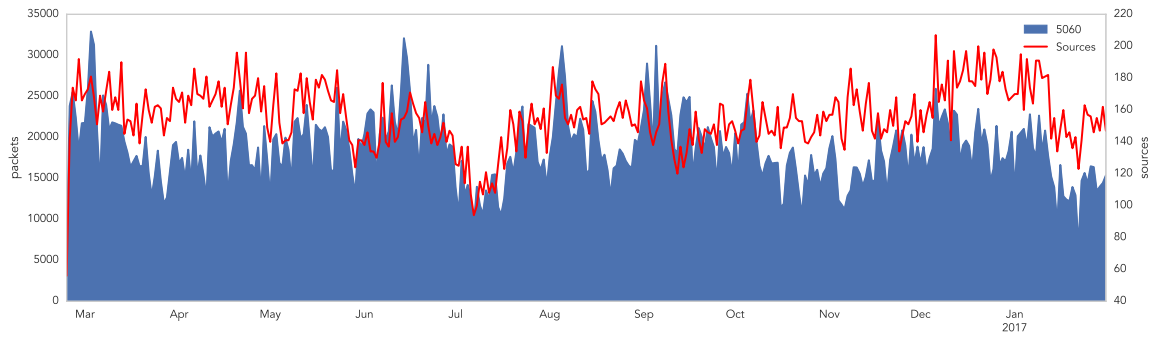


Figure 5.14: SIP scan classified traffic daily evolution with number of scanning sources

5.3.3 Web Services Scanner

Web servers are prone to vulnerabilities (see Section 5.2.1.1) such as XSS and SQL injection. Traffic targeting Web applications would account for a large portion if the sensors were covering websites, but scanners which aim at HTTP(S) ports such as TCP/80, 443, 591, 8008, 8080, or 8443 are less present in IBN. Two reasons might explain this phenomenon: (1) IBN carries mostly non-targeted attacks, i.e. with no prior knowledge of the victim, and (2) other vulnerabilities, which are more easily exploitable (see Section 5.2.1), are present at the time of the analysis. Traffic destined to TCP/443, the standard port for HTTPS, might be associated with the trend of encapsulating malware material into Secure Sockets Layer (SSL) traffic to hide malicious actions [13].

5.3.4 Other Vulnerabilities

An information disclosure vulnerability in the Linux kernel's networking subsystem was discovered and published on July 12th 2016 [CVE-2016-5696]. The implementation of the challenge `ACK` rate limiting allowed an attacker to infer both sequence and acknowledgment numbers about a given connection by creating congestion on the global challenge limit counter. Thus, by measuring the changes with probing packets an attacker can hijack an ongoing TCP connection [46]. No clear trend can be noticed on the dataset. The GRE protocol has also suffered a buffer overflow vulnerability on October 16 [CVE-2016-1453] but no clear trend in collected traffic could be remarked.

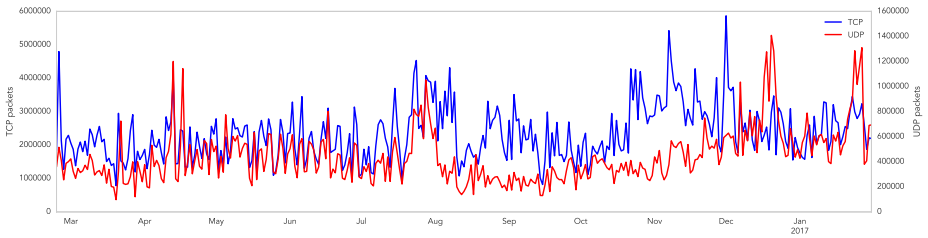
5.4 Discussion

The following subsections will focus on the discussion of the traffic analysis conducted in this chapter. The validity of the overview of the IBN traffic collected at sensors is discussed through different statistics on the number of packets and sources seen across time. The accuracy of global IBN description from the sensor network perspective is analyzed in the following paragraphs in terms of traffic volume, i.e. number of packets, and sources scanning top ports (see Section 5.2.1).

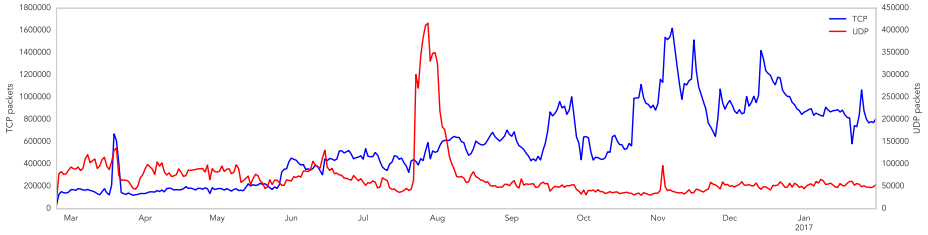
The burst of UDP traffic in July and August 2016 which is visible in Figure 5.15b is related to an increase of sources (see Figure 5.16b) targeting the Netis vulnerability on port 53413 [93]. For the month of July 2016, a unique Swisscom customer address is targeted by more sources than the usual, thus more traffic is observed. Focusing on this sole destination address even if the number of packets increases, the average number of packets by source decreases. Since Netis routers are used by private parties, attackers mainly target ISP's customer ranges (see Section 4.1.1.2). This could explain the difference in volume and number of sources between SANS data and the collected traffic. Indeed, this traffic is not pure IBN because it is more targeted, it reflects attackers' knowledge of their targets. Two destination addresses are mainly targeted by a large number of sources in the beginning of August 2016. These two IP addresses are Open System's firewall devices. Attackers may have inferred that these two IP addresses are used by an operating device because other attempts towards the Open Systems /19 IP range would have resulted into an ICMP `Destination Unreachable` response [11].

5.4.1 Traffic Volume Evolution

IBN traffic volume has dramatically increased during the past year mostly because of the upward trend in IoT botnets number and average size. The compromised machines are constantly scanning the Internet looking for new victims to infect (see Section 5.3.1). Figure 5.15a shows IBN volume evolution in number of packets destined to all top destination ports (see Section 5.2.1) from the SANS Internet Storm Center (ISC) DShield [92] database where firewall users, either private or business users, can share intrusion information. By comparing Figures 5.15a and 5.15b common trends can clearly be identified in terms of packet volume.



(a) SANS ISC DShield TCP/UDP monthly traffic volume evolution

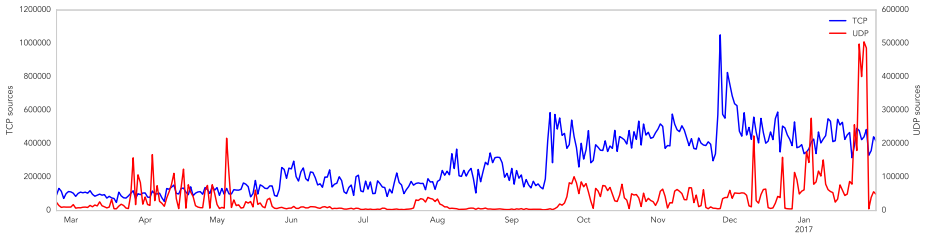


(b) TCP/UDP monthly traffic volume evolution collected by the sensor network

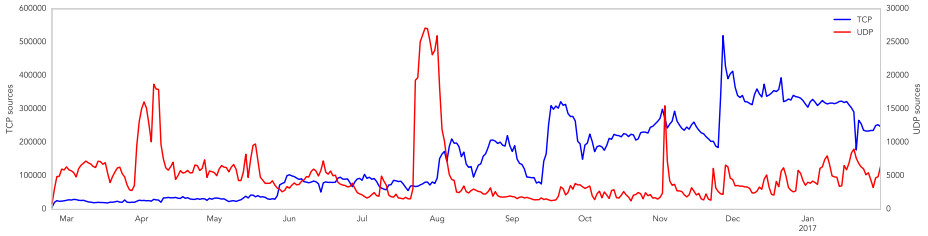
Figure 5.15: Comparison of TCP and UDP traffic volume on all top destination ports (see Section 5.2.1) evolution with SANS data.

5.4.2 Sources Evolution

The number of sources sending IBN was naturally affected by the spreading of worms targeting IoT devices during 2016. The increasing amount of insecure IoT devices and the leakage of the Mirai worm probably is the start of a tendency of increasing number of worms and infected devices. Since there are more connected devices, this results into more potential victims and once infected, they start spreading the worm themselves, thus generate more IBN. Figures 5.16a and 5.16b compare the evolution of the number of sources targeting frequent TCP and UDP ports respectively. This confirms the accuracy of the observed portion of IBN against larger observation databases in terms of the number of sources evolution.



(a) SANS ISC DShield TCP/UDP sources number evolution



(b) TCP/UDP sources number evolution collected by the sensor network

Figure 5.16: Comparison of TCP and UDP sources number on all top destination ports (see Section 5.2.1) evolution with SANS data.

6

Intelligence Extraction System

The information extraction system is presented throughout this chapter. It implements a real-time classification procedure to detect sources of malicious traffic or anomalies to have an overview that would help in possible threat mitigation procedures. The main purpose of the system is to (1) provide a medium-term overview of the trends in Internet Background Noise (IBN) and (2) generate categorized blacklists of Internet Protocol (IP) addresses that behave suspiciously.

6.1 System Architecture

Figure 6.1 gives a description of the system's architecture. Three main actions have to be performed before IBN traffic can be analyzed. At first, denied traffic is logged through `netfilter` and `iptables` (see Sections 4.1.1.3 and 4.1.1.4). These logs are sent to a single collector machine called a log server using `syslog` (see Section 4.1.1.5). Finally, making use of a Python script and a regular expression, logs are parsed and merged (see Section 4.1.4).

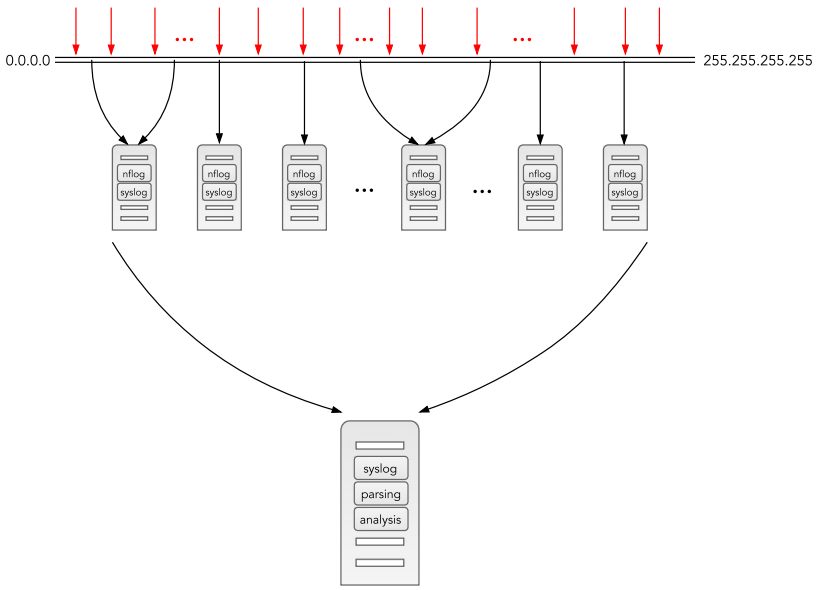


Figure 6.1: Log collection and analysis system overview. Sensors gather IBN by monitoring parts of the IPv4 address space. Packet headers are centralized in the `syslog` server for parsing and analysis.

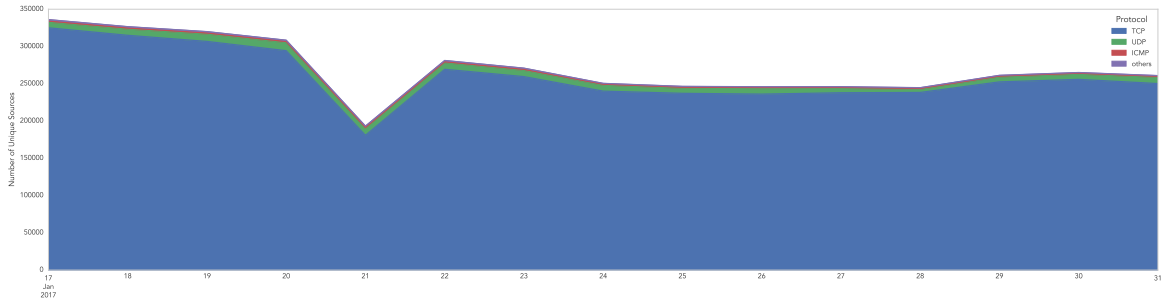
The aggregation system is divided into two independent parts. The first one is intended to analyze IBN general trends, i.e. provide statistics from medium-term aggregations of the IBN traffic using only a subset of representative hosts. The other one is focused on generating blacklists of malicious classified sources observed on a short-term basis. Both subsystems have parameterizable analysis time window and refresh rate. The analysis time window, i.e. how long in the past the analysis takes place, and the refresh rate which is the interval between two analysis are user inputted parameters. Nevertheless, default values are provided in the configuration file.

6.2 Traffic Analysis

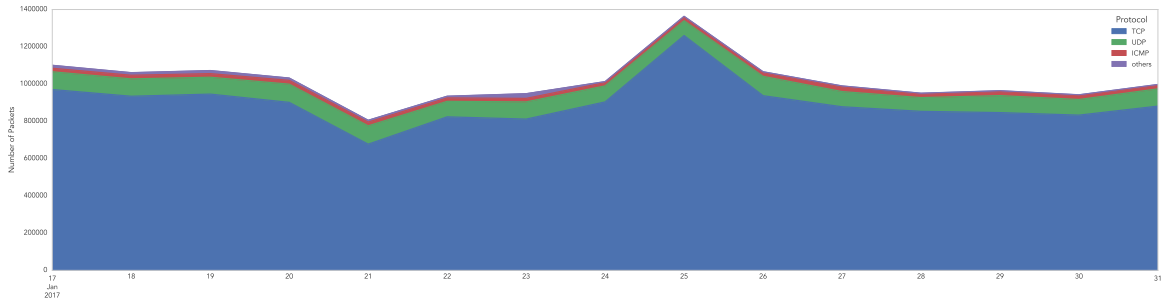
As logs are aggregated on the collector, traffic denied at sensor hosts is selected to generate statistics that will provide an overview of the past and ongoing IBN activity. The evolution of the number of unique sources generating IBN traffic (see Figure 6.2a), packet size distribution (see Figure 6.2c), and protocol portions (see Figure 6.2b) in the traffic are computed. Coupled together, these statistics help gaining awareness into activities and behavior classes constituting IBN. An increase of the number of scanning sources might reflect the discovery of a vulnerability in some service or protocol implementation that more scanners would try to exploit. The packet size could then also be affected and show an increase of small probe packets. Consequently, the proportion of the protocol that is touched by the vulnerability growth. Finally, top ports evolution and statistics may reflect the condition with an increase of the rank of the targeted port. In addition to that and for a fine-grained analysis, top destination ports evolution for both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are computed (see Figure 6.3) along with top destination ports combinations in number of sources (see Table 6.1).

	Sources	Percentage
[TCP/23, TCP/2323]	191001	11.10%
[TCP/23, TCP/7547]	126432	7.35%
[TCP/23, TCP/5358]	97809	5.68%
[TCP/22, TCP/23]	30772	1.79%
[TCP/2323, TCP/5358]	21189	1.23%
[TCP/23, TCP/2323, TCP/5358]	20722	1.20%
[TCP/22, TCP/2222]	20052	1.17%
[TCP/23, TCP/2222]	12281	0.71%
[TCP/23, TCP/6789]	11638	0.68%
[TCP/22, TCP/23, TCP/2222]	11235	0.65%

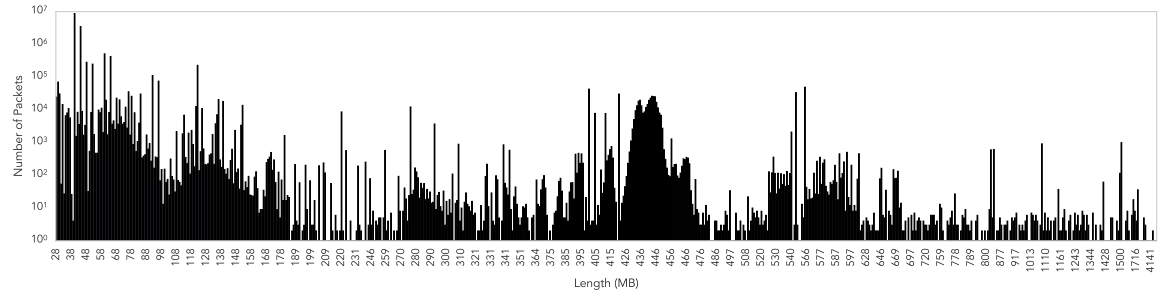
Table 6.1: Top destination port combinations by number of unique sources on a medium-term basis from January 17th to January 31st, 2017. Note that the rows are not mutually exclusive as one source may combine multiple ports vectors.



(a) Evolution of the number of sources generating IBN traffic on a medium-term basis

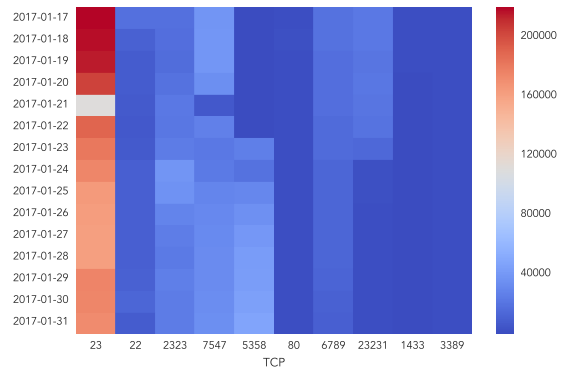


(b) Evolution of protocol portions in IBN traffic on a medium-term basis

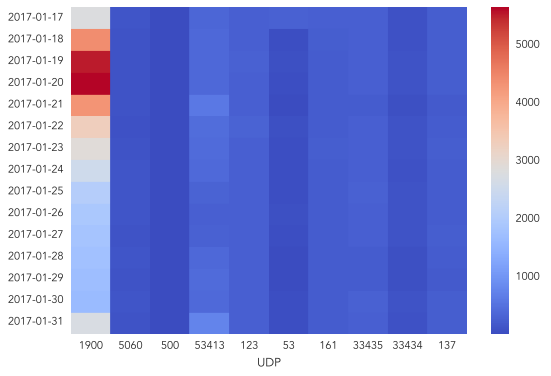


(c) Size distribution of packets populating IBN on a medium-term basis

Figure 6.2: IBN traffic evolution on a medium-term basis



(a) Daily evolution of the number of unique sources targeting the top 10 TCP destination ports in IBN.



(b) Daily evolution of the number of unique sources targeting the top 10 UDP destination ports in IBN.

Figure 6.3: Daily evolution of the number of unique sources targeting the top 10 TCP and UDP destination ports in IBN on a medium-term basis as a waterfall heatmap. Ports are sorted from left to right in descending order by number of sources in total.

6.3 Sources Classification

This section presents the source classification system used to generate blacklists of suspicious IP addresses by analyzing a short time frame of the traffic. The classification is done in three steps: (1) extract basic features for each source, (2) perform a preliminary classification, and (3) compute specific features for each behavior class. Note that this classification scheme is made to be easily extended to take new classes into account.

6.3.1 Source Features Extraction

A preliminary feature extraction allows to differentiate sources according to their behavior during the analyzed time frame. The following values are extracted for each source:

- The number of packets a source sent (`occurrences`)
- The number of unique destinations it sent traffic to (`dst`)
- The number of unique destination ports it used (`dpt`)
- The protocol it uses if it is unique (`proto`)
- The source port it uses if it applies and is unique (`spt`)
- The TCP flag combination it uses if it applies and is unique

6.3.2 Source Classification

Sources related features permit the classification of sources into three main categories: (1) port scanners, (2) misconfigured devices, and (3) sources generating backscatter traffic (see Section 2.3.4). Sources appearing only once during the time frame are not further processed and are classified as one hitters (`ONEHIT`). Figure 6.4 shows a tree abstraction of the source classification scheme. The *time-to-live* of a blacklist entry can be parameterized as a multiple of the analysis time window. However, a newer classification, if not a `ONEHIT`, always has priority over an older one. Depending on categories, additional features are extracted. Table 6.2 shows a summary of the extracted source features.

6.3.2.1 Port Scanners Analysis

According to definitions seen in Section 2.3.2 about port scanning, scanners can be categorized into the three main classes that are namely, horizontal (`HSCAN`), vertical (`VSCAN`) and block (`BSCAN`) scanners. Another subclass, which encompasses sources sending multiple probes destined to the same port of the same destination (`PSCAN`), is also defined.

Interarrival Time (IAT) [12] (see Section 3.2.2) statistics are computed for every type of scanning source. For horizontal and vertical scanners, the IAT, is by definition the time between two probes, however for block scanners it is defined as the time between two probes directed to the same target. Based on these assumptions, the average μ_{IAT} is computed for each scanning source along with the Coefficient of Variation (CV), $c_{v_{IAT}}$, also known as the Relative Standard Deviation (RSD), which is defined as the ratio between the standard deviation σ_{IAT} to the average μ_{IAT}. CV is a measure of spread for a set of data originally proposed as a way of comparing the variability in different distributions [28]. μ_{IAT} gives indications about the scanning speed of the source while $c_{v_{IAT}}$ helps to know how the scanning speed varies and compare sources with different μ_{IAT} in terms of speed variation of scans. The time span of each source is also computed, i.e. the time difference between the first probe and the last one in the analysis time window, and compared

against the length of the analysis time window to gain insights into the spread of the probes in the time window (`timeSpanWindowRatio`).

The coverage ratio for horizontal and block scanners is computed by dividing the number of destinations of a source scanner by the number of unique destinations encountered during the analysis. In the case of a vertical scanner, the targeted destination is extracted.

Frequent destination port combinations targeted by block and vertical scanners are extracted using the FP-Growth algorithm [36] (see Section 2.4.4.3). The support value s to determine frequent destination port combinations has a default value of 10; however, it can be changed by the user. After frequent combinations have been extracted, each scanner classified source is assigned the frequent combination that is the most similar to its destination port vector. Then, a score of similarity with the assigned frequent destination port combination is computed for each source. Grouping sources with similar target destination ports vectors is the first step in detecting botnets.

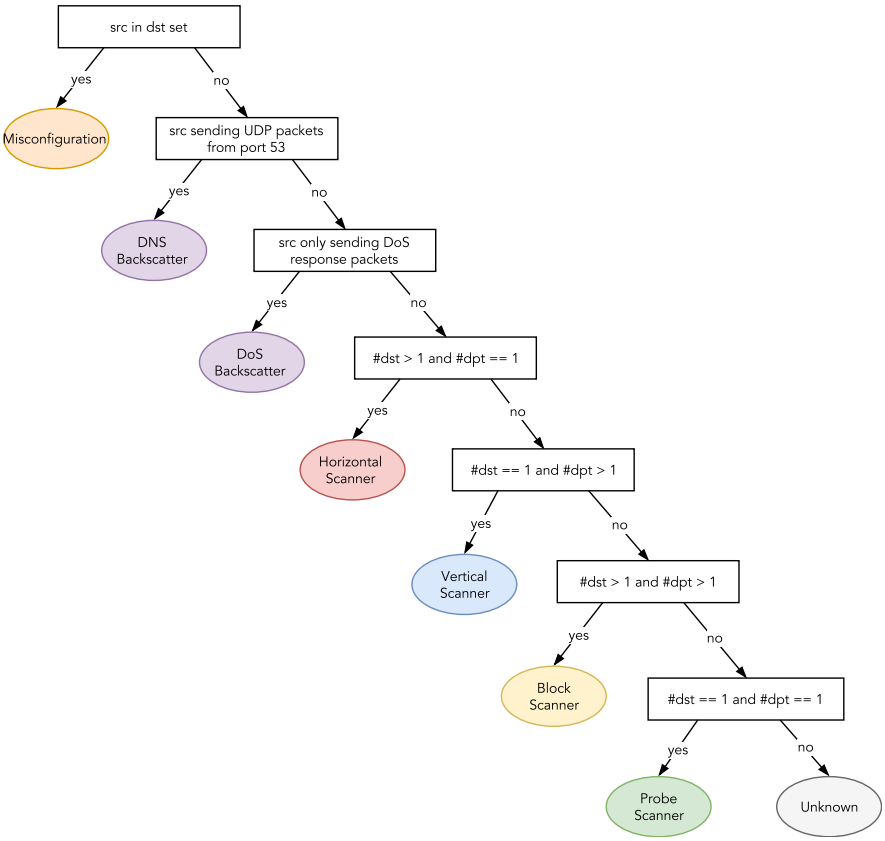


Figure 6.4: Classification tree depicting the categories where each source is classified according to its feature values. Unknown is a default class that encompasses sources sending traffic using different protocols.

6.3.2.2 Misconfigurations Analysis

Traffic coming from encountered destination addresses is classified as traffic caused by a misconfigured or not yet configured device. This can happen when firewall rules that allow specific traffic are not added yet. Sources of such blocked traffic are labeled as misconfigurations (`MISCONFIG`) allowing to better identify them and fix the configurations or update the firewall policies. The list of all IP addresses that blocked a packet coming from `MISCONFIG` labeled sources is extracted to help with adjusting the host configuration. These packets could also be spoofed but it is unlikely because it would require the attacker to know which IP addresses are used for managing which device. Additionally, backscatter traffic would have been gathered at the spoofed device.

6.3.2.3 Backscatter Analysis

Backscatter traffic is generated by sources in response to IBN traffic. A source sending traffic with characteristic TCP flag combinations are classified as denial of service (DoS) backscatter generators [67] (`DOSBACKSCATTER`) and may be under a DoS attack (see Section 5.1.1.1 and Table 5.2). Source sending traffic from port UDP/53 may also be a generator of Domain Name Service (DNS) backscatter traffic (`DNSBACKSCATTER`) (see Section 2.3.4.2). Additionally, TCP flags combinations are extracted for each source sending DoS backscatter traffic.

Feature
Occurrences
Number of Destination Ports
Number of Destinations
Protocol
Source Port
TCP Flags
μ_{IAT}
$c_{v_{IAT}}$
Time Span Ratio
Destination Coverage
Frequent Destination Ports Set
Similarity with Frequent Destination Port Set
Destinations

Table 6.2: Source features by classes. Black cells indicate features that are always extracted while gray ones represent features that are only extracted if the source always uses the same values for each sent packet.

6.3.3 Classes Size Evolution

Statistics are computed on the evolution of classes proportion on a medium-term basis to get insights into the apparition of new scanners and the changes in classifications. Figure 6.5 shows a visualization of these statistics.

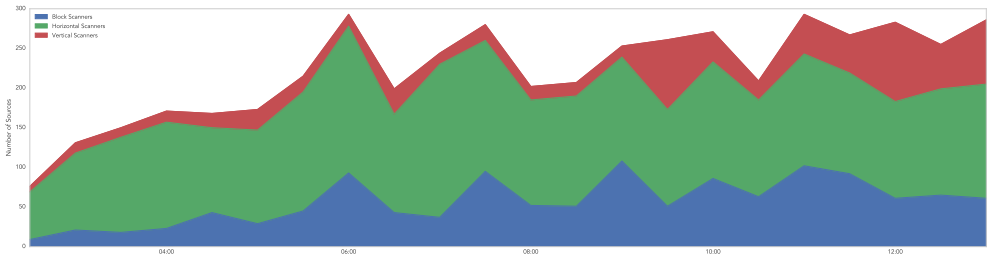


Figure 6.5: Evolution of the proportion of newly classified scanners for a two and a half hours time window and a refresh rate of 30 minutes on February 26th, 2016.

6.4 Discussion

Various data analysis methods have been tested during the exploratory phase. *t-Distributed Stochastic Neighbor Embedding (t-SNE)* [60] was used to reduce the dimension of aggregations and visualize the data in a two or three-dimensional space. Statistics such as the number of packets a specific source sent or the number of different destination ports it used (see Section 6.3.1) have been computed, reduced and visualized (see Figure 2.5a) with the help of the t-SNE implementation available in the Scikit-Learn library (see Section 4.2.2.5). DBSCAN [26], a density-based clustering algorithm, was also used to group sources with similar behavior (see Section 2.4.4.1). Grouping similar behaving sources in near real-time is related to botnet detection, which would require more computation time, a broader time frame of observation and data from more sensors. Thus, a trade-off is to be found between real-time and computation time, which is related to the size of the data, to get satisfactory results (see Section 7.4). Data aggregation can be challenging, e.g. when a summary of destination ports for each source has to be mapped to a number for inputting the data into a clustering algorithm.

7

Future Work

Research opportunities following this project are outlined in this chapter. This project is a proof of concept, several works which can be derived from the thesis and follow-up work ideas are described in the following subsections.

7.1 Internet Background Noise Traffic Investigator

Several aggregations by source IP address are computed in the scope of the blacklist generation process. However, the extracted statistics might not perfectly detect malicious attacks with a more complex behavior. A tool implementing a specific grammar that would permit to dig deeper in the raw logs of the denied traffic would be valuable. This grammar could act as an interface on top of Pandas (see Section 4.2.2.2), the Python data analysis library, exploiting its SQL-like aggregation capabilities.

7.2 Multi-layer Aggregation Architecture

The analysis was conducted on a subset of hosts. A global deployment would introduce scaling challenges when collecting and correlating data from a much larger number of sensors. A solution to this problem could be to scale out by devising a layered architecture of aggregators where each aggregator of the first layer would receive the data from a subset of sensors. The second layer would then gather the aggregations made by the first layer and merge them using a Map-Reduce [22] approach with efficient probabilistic, hash-based data structures such as Bloom [8] or Cuckoo [29] filters.

7.3 Interactive Blacklist and Visualizations

Statistics and time series are computed periodically over the past received traffic. Nevertheless, no visualizations are generated for the computed data. Combined with the Internet Background Noise (IBN) Traffic Investigator (see Section 7.1), a widget providing statistics and visualizations that would generate relevant visualizations in the existing Mission Control™ Portal used by both customers and engineers would be beneficial to gain insights into the evolution of IBN.

7.4 Botnet Detection

A similar approach to the one described in Section 6.4 where sources are clustered by their feature vectors could be developed. Similar sources, i.e. sources showing the same behavior, would be grouped so that botnets can be identified providing a confidence index for every discovered botnet. The feature changes of such sources could be tracked to study the evolution of the botnet. For example, a future work could detect if it targets new destination ports by extracting the most frequent destination ports combination of each cluster of sources (see Figure 2.5a) and keep track of the previous combinations. To measure a scanner from an outside perspective and try to infer if its behavior evolves, the Relative Uncertainty (RU) of its destination port vector can be computed (see Section 2.4.5) and compared with the previous value to compare the uniqueness of destination port values between both previous and current distributions. A large difference between the previous and the current RU values could indicate a behavior change and would help in predicting further evolution. By focusing on destination IP addresses targeted by each botnet, it may be possible to establish a cartography of botnets and group them by Command and Control (C&C) server. A botnet that is well organized will less likely scan the same destination more than once on the same port. Consequently, this would help in differentiating sources behaving similarly that are part of different botnets.

7.5 IPv6 Internet Background Noise Analysis

This work describes and analyzes IP version 4 IBN traffic, however, IP version 6 IBN traffic will increase as IPv6 is becoming more used. Researchers studied IPv6 IBN [19, 43, 82]. In every observation, a low level of traffic was observed and seemed to be entirely attributable to misconfigurations. No traffic could be attributed to systematic scan of the IPv6 address space [43]. This can be explained by the sparse population of the IPv6 address space [19, 43, 82]. The system could be extended to take IPv6 traffic into account for hosts that implement the protocol. This approach would be valuable to gain insights into the evolution of IPv6 malicious IBN by having less but qualitative data to analyze.

8

Conclusion

In this master thesis on Extracting Cyber Threat Intelligence from the Internet Background Noise, an analysis of today's Internet Background Noise (IBN) on denied IPv4 traffic at Open Systems hosts is first made with an additional focus on frequent patterns through several case studies. Data analysis techniques are used to generate an overview of the current IBN trends and provide blacklists of suspicious IP addresses.

Two main challenges during the data exploratory part of this work have been faced. At first, extracting relevant features for each source class involved a lot of research in the cyber security area trying to think from an attacker's point of view. Then, it was concluded that machine learning and clustering techniques are not ready-to-use solutions to all problems. Besides attacking techniques being in constant evolution, network traffic data is more heterogeneous than image data for example. These methods might work well for widespread uses such as photo of text recognition, but their application to real-world network security issues is at an early stage.

Other research works analyzing IBN state about rises of IBN traffic during the preceding years because of worms spreading was done, but the analysis of the current IBN revealed an unprecedented increase by a factor of four of such traffic over the past year alone due to the rise of Internet of Things (IoT) botnets. An extensible system extracting security relevant information from IBN such as general trends in IBN and blacklists of malicious source IPs has also been implemented. Since this system solely relies on standardized `iptables`, `netfilter` and `syslog` protocol, it is applicable to other setups.

Once reviewed and thoroughly tested, the prototype will be deployed in the production environment of Open Systems AG and thus provide security engineers with relevant information, based upon which they may easily correlate multi-staged events to further improve Mission Control™ Security Services.

Appendix

List of Figures

List of Tables

List of Acronyms

AS	Autonomous System.
BGP	Border Gateway Protocol.
C&C	Command and Control.
CIFS	Common Internet File System.
CIP	Common Industrial Protocol.
CV	Coefficient of Variation.
DDoS	Distributed Denial of Service.
DMZ	Demilitarized Zone.
DNS	Domain Name Service.
DoS	Denial of Service.
DSL	Digital Subscriber Line.
ESP	Encapsulating Security Payload.
FIM	Frequent Itemset Mining.
FTP	File Transfer Protocol.
GRE	Generic Routing Encapsulation.
HTTP	Hypertext Transfer Protocol.
IANA	Internet Assigned Number Authority.
IAT	Interarrival Time.
IBN	Internet Background Noise.
IBR	Internet Background Radiation.
ICMP	Internet Control Message Protocol.
IDS	Intrusion Detection System.
IGMP	Internet Group Management Protocol.
IoT	Internet of Things.
IP	Internet Protocol.
IRC	Internet Relay Chat.
ISC	Internet Storm Center.
ISP	Internet Service Provider.
JSON	JavaScript Object Notation.
LAN	Local Area Network.
NAT	Network Address Translation.
NFLOG	Netfilter Log.
NIDS	Network Intrusion Detection System.
NTP	Network Time Protocol.

OS Operating System.
P2P Peer-to-peer.
PAT Port Address Translation.
rDNS reverse DNS.
RDP Remote Desktop Protocol.
RSD Relative Standard Deviation.
RSVP Resource Reservation Protocol.
RU Relative Uncertainty.
SCTP Stream Control Transmission Protocol.
SIP Session Initiation Protocol.
SMB Server Message Block.
SMTP Simple Mail Transfer Protocol.
SNMP Simple Network Management Protocol.
SQL Structured Query Language.
SSDP Simple Service Discovery Protocol.
SSH Secure Shell.
SSL Secure Sockets Layer.
TCP Transmission Control Protocol.
TRW Threshold Random Walk.
TTL Time To Live.
UDP User Datagram Protocol.
VNC Virtual Network Computing.
VoIP Voice over IP.
VPN Virtual Private Network.
WAN Wide Area Network.
XSS Cross-site Scripting.

References

[1] CVE Details NTP vulnerabilities list. https://www.cvedetails.com/vulnerability-list/vendor_id-2153/NTP.html, 2017. [Cited on page 37.]

[2] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994. [Cited on page 11.]

[3] Mark Allman, Vern Paxson, and Jeff Terrell. A brief history of scanning. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 77–82. ACM, 2007. [Cited on page 4.]

[4] Anna-senpai. Mirai source code. https://github.com/jgamblin/Mirai-Source-Code, 2016. [Cited on pages 41, 42, and 43.]

[5] Michael Bailey, Evan Cooke, Farnam Jahanian, and David Watson. The blaster worm: Then and now. *IEEE Security & Privacy*, 3(4):26–31, 2005. [Cited on page 18.]

[6] Chandrajit Bajaj and C Fl John Wiley. Data visualization techniques. 1998. [Cited on page 9.]

[7] Karyn Benson, Alberto Dainotti, Alex C Snoeren, Michael Kallitsis, et al. Leveraging internet background radiation for opportunistic network analysis. In *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, pages 423–436. ACM, 2015. [Cited on pages 4, 15, 20, 25, and 26.]

[8] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970. [Cited on page 57.]

[9] Christian Borgelt. Frequent item set mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6):437–456, 2012. ISSN 1942-4795. doi: 10.1002/widm.1074. [Cited on page 29.]

[10] Christian Borgelt. PyFIM frequent item set mining for python, 2012. URL http://www.borgelt.net/pyfim.html. [Cited on page 29.]

[11] S. Bradner and V. Paxson. IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers. RFC 2780 (Best Current Practice), March 2000. URL http://www.ietf.org/rfc/rfc2780.txt. Updated by RFCs 4443, 5237, 5771, 6335, 7045. [Cited on pages 28, 33, and 46.]

[12] Nevil Brownlee. One-way traffic monitoring with iatmon. In *International Conference on Passive and Active Network Measurement*, pages 179–188. Springer, 2012. [Cited on pages 19, 20, and 52.]

[13] J Michael Butler. Finding hidden threats by decrypting ssl. *SANS Institute*, 2013. [Cited on page 44.]

[14] B. Carpenter and L. Lynch. BCP 101 Update for IPR Trust. RFC 4371 (Best Current Practice), January 2006. URL http://www.ietf.org/rfc/rfc4371.txt. [Cited on page 16.]

[15] D Brent Chapman and Elizabeth D Zwicky. Building internet firewalls. 1999. [Cited on page 3.]

[16] Roger Christopher. Port scanning techniques and the defense against them. Technical report, SANS Institute, 2001. [Cited on page 5.]

[17] M. Cotton, L. Eggert, J. Touch, M. Westerlund, and S. Cheshire. Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry. RFC 6335 (Best Current Practice), August 2011. URL `http://www.ietf.org/rfc/rfc6335.txt`. [Cited on page 5.]

[18] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012. [Cited on page 13.]

[19] Jakub Czyz, Kyle Lady, Sam G Miller, Michael Bailey, Michael Kallitsis, and Manish Karir. Understanding ipv6 internet background radiation. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 105–118. ACM, 2013. [Cited on page 58.]

[20] Alberto Dainotti, Roman Amman, Emile Aben, and Kimberly C Claffy. Extracting benefit from harm: using malware pollution to analyze the impact of political and geophysical events on the internet. *ACM SIGCOMM Computer Communication Review*, 42(1):31–39, 2012. [Cited on pages 4, 8, 20, 25, and 26.]

[21] Alberto Dainotti, Alistair King, Ferdinando Papale, Antonio Pescape, et al. Analysis of a/0 stealth scan from a botnet. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 1–14. ACM, 2012. [Cited on pages 20, 25, 26, and 44.]

[22] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008. [Cited on page 57.]

[23] Roland Dobbins. Mirai iot botnet description and ddos attack mitigation. *Arbor Threat Intelligence*, 28, 2016. [Cited on pages 35 and 42.]

[24] Alexandre Dulaunoy, Gérard Wagener, Marc Stiefer, and Cynthia Wagner. The void–an interesting place for network security monitoring. 2014. [Cited on pages 7, 8, 25, and 26.]

[25] Levent Ertoz, Eric Eilertson, Aleksandar Lazarevic, P Tan, Paul Dokas, Jaideep Srivastava, and V Kumar. Detection and summarization of novel network attacks using data mining. *Minnesota INtrusion Detection System (MINDS) Technical Report*, 2003. [Cited on page 19.]

[26] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996. [Cited on page 55.]

[27] Loras R Even. Honey pot systems explained. *Retrieved May*, 20:2005, 2000. [Cited on page 17.]

[28] Brian S Everitt. *The Cambridge dictionary of statistics*. Cambridge University Press, 2006. [Cited on page 52.]

[29] Bin Fan, Dave G Andersen, Michael Kaminsky, and Michael D Mitzenmacher. Cuckoo filter: Practically better than bloom. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 75–88. ACM, 2014. [Cited on page 57.]

[30] The Shadowserver Foundation. The Shadowserver Foundation vulnerable netis router scanning project, 2017. URL `https://netisscan.shadowserver.org/`. [Cited on page 36.]

[31] The Shadowserver Foundation. The Shadowserver Foundation open resolver scanning project, 2017. URL `https://dnsscan.shadowserver.org/`. [Cited on page 37.]

[32] Kensuke Fukuda and John Heidemann. Detecting malicious activity with dns backscatter (extended). 2015. [Cited on pages 7 and 20.]

[33] Nobuaki Furutani, Tao Ban, Junji Nakazato, Jumpei Shimamura, Jun Kitazono, and Seiichi Ozawa. Detection of ddos backscatter based on traffic features of darknet tcp packets. In *Information Security (ASIA JCIS), 2014 Ninth Asia Joint Conference on*, pages 39–43. IEEE, 2014. [Cited on page 20.]

[34] R. Gerhards. The Syslog Protocol. RFC 5424 (Proposed Standard), March 2009. URL `http://www.ietf.org/rfc/rfc5424.txt`. [Cited on page 25.]

[35] Eduard Glatz and Xenofontas Dimitropoulos. Classifying internet one-way traffic. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 37–50. ACM, 2012. [Cited on pages 4, 10, 15, 18, 19, 20, 25, and 26.]

[36] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM Sigmod Record*, volume 29, pages 1–12. ACM, 2000. [Cited on pages 11, 12, 29, and 53.]

[37] Michael Heller. More ddos dns amplification attacks use ssdp than ntp, 2016. URL `http://searchsecurity.techtarget.com/news/450302279/More-DDoS-DNS-amplification-attacks-use-SSDP-than-NTP`. [Cited on page 36.]

[38] Ben Herzberg, Dima Bekerman, and Igal Zeifman. Incapsula breaking down mirai: An iot ddos botnet analysis, 2016. URL `https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html`. [Cited on page 42.]

[39] James Hoagland. The teredo protocol: Tunneling past network security and other security implications. Technical report, SANS Institute, 2000. [Cited on page 36.]

[40] honeynet Project. Know your enemy: Honeynets, 2006. URL `http://old.honeynet.org/papers/honeynet/`. [Cited on page 17.]

[41] Darrell Huff. *How to lie with statistics*. WW Norton & Company, 2010. [Cited on page 9.]

[42] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science Engineering*, 9(3):90–95, May 2007. ISSN 1521-9615. [Cited on page 29.]

[43] Geoff Huston. Background radiation in ipv6. *The ISP Column, APNIC*, 2010. [Cited on page 58.]

[44] IBM. IBM® X-Force Exchange: Cloud-based, threat intelligence sharing platform. URL `https://exchange.xforce.ibmcloud.com/`. [Cited on page 32.]

[45] Félix Iglesias and Tanja Zseby. Entropy-based characterization of internet background radiation. *Entropy*, 17:74–101, 2015. [Cited on pages 10 and 19.]

[46] Akamai InfoSec. Vulnerability in the linux kernel's tcp stack implementation, 2016. URL `https://blogs.akamai.com/2016/08/vulnerability-in-the-linux-kernels-tcp-stack-implementation.html`. [Cited on page 45.]

[47] Yu Jin, György Simon, Kuai Xu, Zhi-Li Zhang, and Vipin Kumar. Gray's anatomy: Dissecting scanning activities using ip gray space analysis. In *Proceedings of the 2nd USENIX workshop on Tackling computer systems problems with machine learning techniques*, pages 1–6. USENIX Association, 2007. [Cited on pages 18, 20, 25, and 26.]

[48] Yu Jin, Zhi-Li Zhang, Kuai Xu, Feng Cao, and Sambit Sahu. Identifying and tracking suspicious activities through ip gray space analysis. In *Proceedings of the 3rd annual ACM workshop on Mining network data*, pages 7–12. ACM, 2007. [Cited on pages 18, 20, 25, and 26.]

[49] Alan Jones. Netfilter and iptables–a structural examination. *SANS Institute Reading Room site*, 2004. [Cited on page 23.]

[50] M. Joseph and J. Susoy. P6R's Secure Shell Public Key Subsystem. RFC 7076 (Informational), November 2013. URL `http://www.ietf.org/rfc/rfc7076.txt`. [Cited on page 35.]

[51] Jaeyeon Jung, Vern Paxson, Arthur W Berger, and Hari Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, pages 211–225. IEEE, 2004. [Cited on pages 19 and 25.]

[52] Kris Katterjohn. Port scanning techniques, 2007. [Cited on pages 32 and 33.]

[53] Brian Krebs. KrebsOnSecurity hit with record ddos, 2016. URL `https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/`. [Cited on page 43.]

[54] Brian Krebs. KrebsOnSecurity source code for iot botnet 'mirai' released, 2016. URL `https://krebsonsecurity.com/2016/10/source-code-for-iot-botnet-mirai-released/`. [Cited on page 41.]

[55] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach (6th Edition)*. Pearson, 6th edition, 2012. ISBN 0132856204, 9780132856201. [Cited on page 4.]

[56] Cynthia Bailey Lee, Chris Roedel, and Elena Silenok. Detection and characterization of port scan attacks. *Univeristy of California, Department of Computer Science and Engineering*, 2003. [Cited on page 5.]

[57] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2014. [Cited on page 10.]

[58] Christopher Low. Icmp attacks illustrated. *SANS Institute*, 2001. URL `https://www.sans.org/reading-room/whitepapers/threats/icmp-attacks-illustrated-477`. [Cited on pages 6 and 33.]

[59] Gordon Fyodor Lyon. *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, 2009. [Cited on pages 5 and 31.]

[60] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. [Cited on page 55.]

[61] G. Malkin. Traceroute Using an IP Option. RFC 1393 (Historic), January 1993. URL `http://www.ietf.org/rfc/rfc1393.txt`. Obsoleted by RFC 6814. [Cited on page 6.]

[62] A.M. McKenzie. Telnet Protocol specifications. RFC 495, May 1973. URL `http://www.ietf.org/rfc/rfc495.txt`. Updated by RFC 562. [Cited on page 35.]

[63] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51–56, 2010. [Cited on page 29.]

[64] Wes McKinney. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.", 2012. [Cited on page 29.]

[65] Lihua Miao, Wei Ding, and Haiting Zhu. Extracting internet background radiation from raw traffic using greynet. In *2012 18th IEEE International Conference on Networks (ICON)*, pages 370–375. IEEE, 2012. [Cited on page 18.]

[66] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver. Inside the slammer worm. *IEEE security and privacy*, 1(4):33–39, 2003. [Cited on pages 4 and 20.]

[67] David Moore, Colleen Shannon, Douglas J Brown, Geoffrey M Voelker, and Stefan Savage. Inferring internet denial-of-service activity. *ACM Transactions on Computer Systems (TOCS)*, 24(2):115–139, 2006. [Cited on pages 4, 7, 20, 32, 33, and 54.]

[68] Vagishwari Nagaonkar and John Mchugh. Revisiting the threshold random walk scan detector. *Proceedings of FloCon*, 2008. [Cited on pages 19 and 20.]

[69] Netlab. Netlab mirai scanner, 2016. URL http://data.netlab.360.com/mirai-scanner. [Cited on page 43.]

[70] James P Owens Jr. *A study of passwords and methods used in brute-force SSH attacks*. PhD thesis, Clarkson University, 2008. [Cited on page 35.]

[71] D Lakshmi Padmaja and B Vishnuvardhan. Comparative study of feature subset selection methods for dimensionality reduction on scientific data. In *Advanced Computing (IACC), 2016 IEEE 6th International Conference on*, pages 31–34. IEEE, 2016. [Cited on page 10.]

[72] 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*, pages 27–40. ACM, 2004. [Cited on pages 4, 8, 15, 16, 17, 18, 19, 25, and 26.]

[73] Ignasi Paredes-Oliva, Ismael Castell-Uroz, Pere Barlet-Ros, Xenofontas Dimitropoulos, and Josep Sole-Pareta. Practical anomaly detection based on classifying frequent traffic patterns. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 49–54. IEEE, 2012. [Cited on pages 25 and 26.]

[74] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. [Cited on page 29.]

[75] Hanspeter Pfister, Joe Blitzstein, and Verena Kaynig. Lecture slides in data science (cs109). 2015. [Cited on page 9.]

[76] J. Postel. User Datagram Protocol. RFC 768 (INTERNET STANDARD), August 1980. URL http://www.ietf.org/rfc/rfc768.txt. [Cited on page 34.]

[77] J. Postel. Internet Protocol. RFC 791 (INTERNET STANDARD), September 1981. URL http://www.ietf.org/rfc/rfc791.txt. Updated by RFCs 1349, 2474, 6864. [Cited on page 34.]

[78] J. Postel. Internet Control Message Protocol. RFC 792 (INTERNET STANDARD), September 1981. URL http://www.ietf.org/rfc/rfc792.txt. Updated by RFCs 950, 4884, 6633, 6918. [Cited on pages 6 and 34.]

[79] J. Postel. Transmission Control Protocol. RFC 793 (INTERNET STANDARD), September 1981. URL http://www.ietf.org/rfc/rfc793.txt. Updated by RFCs 1122, 3168, 6093, 6528. [Cited on page 34.]

[80] T. Richardson and J. Levine. The Remote Framebuffer Protocol. RFC 6143 (Informational), March 2011. URL http://www.ietf.org/rfc/rfc6143.txt. [Cited on page 35.]

[81] James Romanski. Using snmp for reconnaissance. *Intrusion Detection FAQ*, 2000. URL http://www.sans.org/newlook/resources/IDFAQ/SNMP.html. [Cited on page 37.]

[82] John Ronan, Matthew Ford, and Jonathan Stevens. Initial results from an ipv6 darknet. 2006. [Cited on page 58.]

[83] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. URL http://www.ietf.org/rfc/rfc3261.txt. Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141, 6665, 6878, 7462, 7463. [Cited on page 44.]

[84] Paul Russell. Linux 2.4 packet filtering howto. 2002. URL http://netfilter.samba.org/. [Cited on page 24.]

[85] David Salomon. *Elements of computer security*. Springer Science & Business Media, 2010. [Cited on pages 3, 4, 5, 7, and 17.]

[86] Salvatore Sanfilippo. Redis open letter to white hat hackers changing passwords of open redis instances, 2015. URL https://gist.github.com/antirez/12d60c950fbfd8c8837e#file-wh-md. [Cited on page 35.]

[87] Akihiro Shimoda and Shigeki Goto. Virtual dark ip for internet threat detection. In *APAN Network Research Workshop*, pages 17–23, 2007. [Cited on pages 4, 8, 15, 17, 18, 25, and 26.]

[88] Dafydd Stuttard and Marcus Pinto. *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws*. John Wiley & Sons, 2011. [Cited on page 35.]

[89] Dennis Sun. Lecture slides in analysis of spatial and temporal data (stats 253). 2015. [Cited on page 9.]

[90] KoonYaw Tan. IDFAQ: How can attacker use icmp for reconnaissance? *SANS Institute*. URL https://www.sans.org/security-resources/idfaq/how-can-attacker-use-icmp-for-reconnaissance/3/13. [Cited on pages 6 and 33.]

[91] Joanne Treurniet. A network activity classification schema and its application to scan detection. *IEEE/ACM Transactions on Networking*, 19(5):1396–1404, 2011. [Cited on page 19.]

[92] Johannes Ulrich. SANS Internet Storm Center a global cooperative cyber threat / internet security monitor and alert system., 2000. URL https://www.dshield.org/. [Cited on page 46.]

[93] Johannes Ulrich. Surge in exploit attempts for netis router backdoor (udp/53413), 2016. URL https://isc.sans.edu/forums/diary/Surge+in+Exploit+Attempts+for+Netis+Router+Backdoor+UDP53413/21337/. [Cited on pages 36 and 46.]

[94] Risto Vaarandi and Kārlis Podiņš. Network ids alert classification with frequent itemset mining and data clustering. In *2010 International Conference on Network and Service Management*, pages 451–456. IEEE, 2010. [Cited on page 19.]

[95] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011. [Cited on page 29.]

[96] Guido Van Rossum and Fred L Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995. [Cited on page 29.]

[97] 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*, pages 62–74. ACM, 2010. [Cited on pages 4, 9, 15, 16, 17, 18, 19, 25, and 26.]

[98] Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 169–180. ACM, 2005. [Cited on pages 13, 15, 19, 25, and 26.]

[99] Kyle York. Dyn statement on 10/21/2016 ddos attack, 2016. URL http://dyn.com/blog/dyn-statement-on-10212016-ddos-attack/. [Cited on page 43.]

[100] Yang Yu. Badtunnel: How do i get big brother power ? Black Hat USA, 2016. URL https://www.blackhat.com/docs/us-16/materials/us-16-Yu-BadTunnel-How-Do-I-Get-Big-Brother-Power.pdf. [Cited on page 36.]