Journal of Cyber Forensics and Advanced Threat Investigations

CFATI OPEN ACCESS

Volume (), . http://dx.doi.org/xxx-xxx-xxx

Scraping airlines bots: insights obtained studying honeypot data

Elisa Chiapponi[1], Marc Dacier[1], Onur Catakoglu[2], Olivier Thonnard[2], Massimiliano Todisco[1]

Abstract

Airline websites are the victims of unauthorised online travel agencies and aggregators that use armies of bots to scrape prices and flight information. These so-called Advanced Persistent Bots (APBs) are highly sophisticated. On top of the valuable information taken away, these huge quantities of requests consume a very substantial amount of resources on the airlines' websites. In this work, we propose a deceptive approach to counter scraping bots. We present a platform capable of mimicking airlines' sites changing prices at will. We provide results on the case studies we performed with it. We have lured bots for almost 2 months, fed them with indistinguishable inaccurate information. Studying the collected requests, we have found behavioural patterns that could be used as complementary bot detection. Moreover, based on the gathered empirical pieces of evidence, we propose a method to investigate the claim commonly made that proxy services used by web scraping bots have millions of residential IPs at their disposal. Our mathematical models indicate that the amount of IPs is likely 2 to 3 orders of magnitude smaller than the one claimed. This finding suggests that an IP reputation-based blocking strategy could be effective, contrary to what operators of these websites think today.

Notes for Practice

- Many web domains suffer from continuous web scraping by bots. Among those, airlines' websites receive a very large amount of bot requests which increases their expenses without providing any revenue.
- The usual countermeasures are mainly based on browser fingerprinting. They are not always effective against sophisticated bots. We propose a new deception technique based on a honeypot platform.
- The analysis of the collected data provides insights on the botnets ecosystem. Behavioural patterns are found in the payloads sent by the bots, offering new ways to detect them.
- It is a common belief among practitioners that an IP blocking strategy against these bots is deemed to fail due to the sheer volume of IPs at their disposal, in particular thanks to paying professional proxy services. We develop mathematical models, validated by our measurements, that question these assumptions.

Keywords

Scraping bots, Honeypot, Deception, Mathematical Modelling

Submitted: 28/02/2021 — **Accepted:** 12/05/2021 — **Published:**

[1] *Digital security Department, Eurecom, Biot, France. {elisa.chiapponi,marc.dacier,massimiliano.todisco}@eurecom.fr*

[2] *Global Security Operations, Amadeus IT Group, Biot, France. {onur.catakoglu,olivier.thonnard}@amadeus.com*

1. Introduction

The Internet has discovered the existence of botnets and the nuisance they can cause in February 2000 with the early DDoS attacks against Yahoo!, Amazon.com, CNN.com, and other major web sites (Dietrich, Long, & Dittrich, 2000). They have continuously evolved from relatively rudimentary pieces of software to very sophisticated components, perpetrating malicious activities against websites (Li, Azad, Rahmati, & Nikiforakis, 2021). To increase their resilience, bots take advantage of proxy services publicly available on the web, for a fee. Some of them claim to provide access to a pool of several millions of residential IP addresses [1]. An IP blocking solution does not appear to be a viable approach to block them due to the, supposedly, sheer volume of IPs, available all over the world.

A 2019 Imperva report (Imperva, 2019) describes very clearly how badly different industries are impacted by these armies

[1] We prefer not to offer them additional advertisement by listing them here. The authors remain available for further information and address the interested readers to (Mi et al., 2019) for an in-depth study of Residential Proxies as a Service.

of bots. In particular, the airline industry suffers from this phenomenon. In 2017, according to that report, the proportion of bad bots traffic to airlines' websites was 43.9 percent. Almost a third of these bad bots were sophisticated ones, referred to as Advanced Persistent Bots (APBs). APBs can mimic human behaviour, load JavaScript and external assets, tamper with cookies, perform browser automation, give the impression that the mouse is moving on the screen, etc.

These bots are used to gather free information from the airlines' sites about flights and ticket prices, damaging the companies' business models. This explains the explosion of a whole new ecosystem of anti-bot techniques. A recent publication (Amin Azad, Starov, Laperdrix, & Nikiforakis, 2020) has shared insights on the design and implementation of 15 popular commercial anti-bot services. Moreover, Vastel et al. explain in detail how bot detection relies on several different fingerprinting techniques to recognise malicious agents (Vastel, Rudametkin, Rouvoy, & Blanc, 2020).

In this complex scenario, our research aims at:

- Gathering a better understanding of the scraping bots ecosystem,
- Investigating new approaches, alternative and/or complementary to browser fingerprinting, to mitigate bots actions.

To achieve these objectives, we collaborated with a major IT provider for the airline industry which hosts several dozens of airlines' websites. These sites are protected by one of the leading commercial anti-bot services.

We introduce a platform where identified bots are served with inaccurate information, at a cheap cost for the data service provider, so that the attackers are the ones consuming needlessly their resources without any hope for a return on their investment. Moreover, studying the empirical evidence collected in our case studies, we gathered more insights about the scraping ecosystem and we propose a behavioural analysis to possibly enhance bot detection. Finally, using the collected empirical evidence, we investigate the conjecture that an IP blocking strategy will always fail. We reach the conclusion that the situation might not be as bleak as it might seem.

To present our work, the paper is structured as follows. In Section 2, we state the problem at hand and our contributions. Section 3 describes the state of the art regarding scraping bots detection, honeypots, and proxy services. Section 4 proposes an overview of the system in which the honeypot is working, while Section 5 shows the results of the first case study we conducted. While we did not consider this case study a success, it led us to successfully carry out a second one. Section 6 explains the second case study, highlighting the differences between the two attempts. Section 7 studies the reputation of the IP addresses of the second case study, showing that they are likely provided by proxy services. In Section 8 we studied the soundness that botnets have at their disposal millions of IP addresses, taking advantage of the empirical data gathered during the second case study. Thanks to mathematical analyses, we reached the conclusion that this is unlikely to be real. In Section 9, we discuss the results of our work and the lessons we learned during the analyses of the data. Limitations and ideas for future work are examined in Section 10. A conclusion is provided in Section 11.

2. Problem definition and contributions

As explained in an industry-specific report, (Imperva, 2019), web scraping perpetrated through bots affects different industries, among which one of the most hit is the airlines' ecosystem.

Unauthorised online travel agencies and aggregators scrape flight booking information without previous agreements with the airlines. Every processed request involves an expensive real-time computation. The problem is that these malicious actors send requests at a very high frequency. None of these requests directly lead to the buying of a ticket but produce sheer number of requests to be treated. The consequence is that the "look-to-book" ratio, which is the ratio between the number of received requests and the number of booked flights artificially increases. This ratio, among other parameters, influences the price of a seat. Moreover, this creates an unfare situation for their competitors, the genuine authorised online travel agencies and aggregators. They pay fees according to their scraping frequencies to obtain the information through legal channels.

Furthermore, competitive airlines use bot campaigns to obtain "up-to-the-minute market intelligence" (Imperva, 2019). In this way, they can change their prices instantly, stealing customers from competitors and thus creating for them a loss of revenues. For several years, there has been an on going battle. It has lead to a number of legal cases and courts decisions such as Ryanair vs Vtours (Regional Court of Hamburg, 2008; Higher Regional Court of Hamburg, 2009) and American Airlines vs FareChase (Cosby, Donald J., 2003), just to name a few.

To protect themselves from bot traffic, airlines' websites take advantage of anti-bot solutions. An arms race exists between scraping bot makers and anti-bot providers. As soon as a family of bots is identified and blocked, mostly thanks to fingerprinting (Vastel et al., 2020), their bot masters replace them with new ones. Blocking all the IP addresses of identified bots is usually not seen as a viable option because it is usually agreed that the real IP addresses of the bots remain hidden behind a large amount of proxy IP addresses provided by professional services. As shown in (Imperva, 2020), in the past years, more and more IP addresses used by bots were coming from residential ISPs. Indeed, proxy services claim to offer to their customers millions of

residential IP addresses, taking advantage of real user devices. They declare that users agree to route the traffic through their devices in exchange for free services.

The benefits of hiding behind this very large pool of IP addresses is threefold for the web scraping actors. First, linking a scraping campaign to any known organisation is impossible, thus no attribution and legal recourse. Second, the impressive number of frequently changing IP addresses used renders any IP blocking strategy impractical. Third, they can run these campaigns with a very limited amount of powerful machines on their back end without the need of any vast and highly distributed infrastructure.

We developed our work collaborating with a major IT provider for the airline industry, whose websites are protected by one of the leading commercial anti-bot services. Having observed the competing efforts of bots and anti-bot products, we have gained the following insights that led to the work presented in this paper:

- *Blocked bots die*: As soon as a certain type of bot is blocked, the traffic associated with that bot disappears. In other words, the anti-bot will have no match anymore against that signature. This means two things about the bot operators. First, they continuously verify the stealthiness and efficacy of their bots. They do not waste their resources sending requests that do not bear fruits. Second, they can modify their bots extremely rapidly (within minutes, or even seconds) to avoid the detection mechanisms put in place against them.
- *Harnessed information is verified*: If the information provided to a bot, such as a ticket price, is quite different from the real one, once more, the traffic associated with that bot disappears, almost immediately (within minutes). This means two more things about the bot operators. First, they continuously verify the correctness of the information they harness, preventing the poisoning of their database. Second, they deduce from the feeding of incorrect information that their bots are now identified and mute them to become stealthy again.

Blocking the bots fuels the arms race, disrupts their operations for a small amount of time but, after that, renders us blind to the new armies they have formed since the bot operators pay attention and are very reactive. A better approach would be to avoid making the bot operators aware that their bots have been identified without incurring the costs associated with the real computation of a response to their requests. This is only possible by providing them an incorrect answer, yet a plausible one to prevent them from understanding what we do.

The contributions of our work are as follow:

1. We show that bots could craft syntactically correct, yet semantically incorrect queries to detect the likely presence of a honeypot. This technique can be observed in the requests of the first case study.
2. We explain how to provide inaccurate information without being detected by the bot operators, as visible in the second case study, where bots received inaccurate content for 53 days without changing their traffic patterns.
3. We show that behavioural patterns emerge when looking at the bot requests of the case studies. This gives hope for another form of detection, on top of the browser fingerprinting approaches, based on the aggregation of all payloads sent by the distributed bots.
4. In the second case study, we identify a specific class of super stealthy bots that are characterised by an extreme distribution of their activity. Most of the time, they only send a single request per day and per IP, and rarely more than two on the same day.
5. We illustrate how, in the second case study, even receiving few requests per day per IP, IP addresses repeat themselves during the running time of the case study. Moreover, we confirm the usage of proxy services by these bots.
6. Using two distinct approaches, we show that i) IP addresses used by the bots of the second case study are not randomly assigned and that ii) the pool size of IPs they are taken from is *two* to *three* orders of magnitude smaller than what is announced by the proxy websites.
7. We explain how the idea of IP-blocking could be rejuvenated to defeat sophisticated bots.

3. State of the art

Botnets, collections of hosts controlled by a bot, have long been used for nefarious activities, such as scraping web pages of different industries (Imperva, 2020).

Over the years, different countermeasures have been proposed to mitigate the scraping phenomenon, such as browser fingerprinting (Laperdrix, Bielova, Baudry, & Avoine, 2020), serving CAPTCHA (Von Ahn, Blum, Hopper, & Langford, 2003) and implementing an IP blocking strategy (Haque & Singh, 2015), already largely used against spam bots (Jung & Sit, 2004).

Nowadays, website owners usually take advantage of third-party anti-bot services to perform bot management. These commercial solutions analyse the incoming requests to the websites. As described in (Vastel et al., 2020), multiple parameters are collected from the environment in which the request is generated, thanks to fingerprinting. This set of parameters can be used to identify the same actor who launches different requests, potentially from different IP addresses. If a signature is recognised as coming from a bot, the corresponding traffic can be blocked or other mitigation actions can be put in place.

Azad et al, (Amin Azad et al., 2020), propose an empirical analysis of some anti-bot services. Unfortunately, their findings indicate that these solutions are mostly efficient against basic bots but not against the truly sophisticated ones. Indeed, an arms race is taking place between anti-bot services trying to fingerprint bots while these bots are very creative to circumvent the detection. This has led the actors behind the bots to perform only small amounts of requests per IP, with the goal of remaining undetected.

As shown in the Imperva Report 2020 (Imperva, 2020), recent years have witnessed the rise of traffic produced by Advanced Persistent Bots (APBs). These bots produce few requests per IP staying below the rate limits and protecting their reputation. They rely on professional proxy services that make large numbers of IP addresses available for these activities (Ni, 2019). These services claim to have access to tens of millions of residential IPs and to be able to rotate them among the different requests of each client. For these reasons, the report (Imperva, 2020) asserts that IP blacklisting has become "wholly ineffective". Doubtlessly, millions of different IPs cannot be blacklisted all together and e-commerce websites cannot risk blocking requests coming from real customers.

In 2019, Mi et Al. (Mi et al., 2019) proposed the first comprehensive study of Residential IP Proxy as a Service. Even if their methodology has been partially criticised for the fingerprinting process of the devices (Samarasinghe & Mannan, 2019b)(Samarasinghe & Mannan, 2019a), they created a successful infiltration framework that enabled them to study residential proxy services from the inside. They collected 6.18 millions of IPs, of which 95,22% are believed to be residential. Among their findings, it is peculiar to see a discrepancy between the number of IPs claimed by one service (30 millions) and the ones collected by them for the same provider (4 millions using 16 million probings). The authors provide no clear explanation for this gap. Furthermore, it is noteworthy to mention the discovery of two providers using the same pool of IPs, while another one built its network on top of a second one. Applying mathematical modelling we also aim at better understanding the residential IP proxies ecosystem by providing a different viewpoint.

The idea behind our project is to change how bot requests are answered. Instead of serving the bots direct feedback of detection, we want to deceive them, letting them think that they are getting the original content. The idea of deceiving an attacker to win him over is certainly not new. Sun Tzu wrote about it more than 500 years before our era in his treaty on the "art of war"(Tzu, 1971). When it comes to computer security, one of its first incarnations dates back to 1986 with the famous Cuckoo's Egg story with Cliff Stoll. A few years later, in 1992, B. Cheswick told us about his "evening with Berferd" (Cheswick, 1992) and, just after that, W. Venema made it possible for everyone to play with the idea with the creation of TCPWrapper (Venema, 1992). In (Cohen, 2006), Cohen formalises the notion of deception in computers and reviews the early works on honeypots. (Garg & Grosu, 2007) provides a game-theoretical view of these deception approaches.

Since then, *honeypots*, *honeynets* and *honeytokens* (Pouget, Dacier, & Debar, 2003) have received a lot of attention. They exist in all kinds of flavors, low/mid/high interaction (Leita & Dacier, 2008). They are implemented from hardware and driver levels up to the application level. Some are simply collecting information about attackers to learn their *modus operandi* (Pouget & Dacier, 2004; Nicomette, Kaaniche, Alata, & Herrb, 2011) or derive actionable knowledge about them (Thonnard & Dacier, 2008), possibly leading to attack attribution applications (Thonnard, Mees, & Dacier, 2009). Others take an active role in slowing down the attackers. They are then usually called sticky honeypots, tarpits or crawler traps (*LaBrea: "Sticky" Honeypot and IDS*, n.d.; Delong, Filiol, & David, 2019). Web application honeypots, in particular, have received a lot of attention, with, among many others, the following interesting pieces of work (Nunes & Correia, 2010; Catakoglu, Balduzzi, & Balzarotti, 2016; Endicott-Popovsky et al., 2009; Djanali et al., 2014). In particular, when it comes to bot detection, (McKenna, 2016) provides a good survey on the various works that have tried to use web-based honeypots and honey-tokens against them.

Recently, Li et al. (Li et al., 2021) studied bots attacking what they called "honeysites", i.e. non publicly advertised websites used to attract bot traffic. They collected requests, without interacting or trying to deceive the bots. They discovered that 64.37% of the collected bot IP addresses were located in residential IP space. This confirms the belief that bots largely take advantage of residential proxy services to conduct all their malicious activities.

In the past, *Bait&Switch* Honeypots (*The Bait and Switch Honeypot*, n.d.) and the Intrusion Trap Systems (Takemori, Rikitake, Miyake, & Nakao, 2003) already proposed to redirect malicious traffic to a honeypot that mirrors the real site under protection. The limitation of these approaches was the creation of simplistic versions of the real website as honeypots. The main differences, concerning our work, lie in the complexity of the system we have to mimic, the sophistication of the attackers to lure, and our desire to provide plausible, yet inaccurate information.

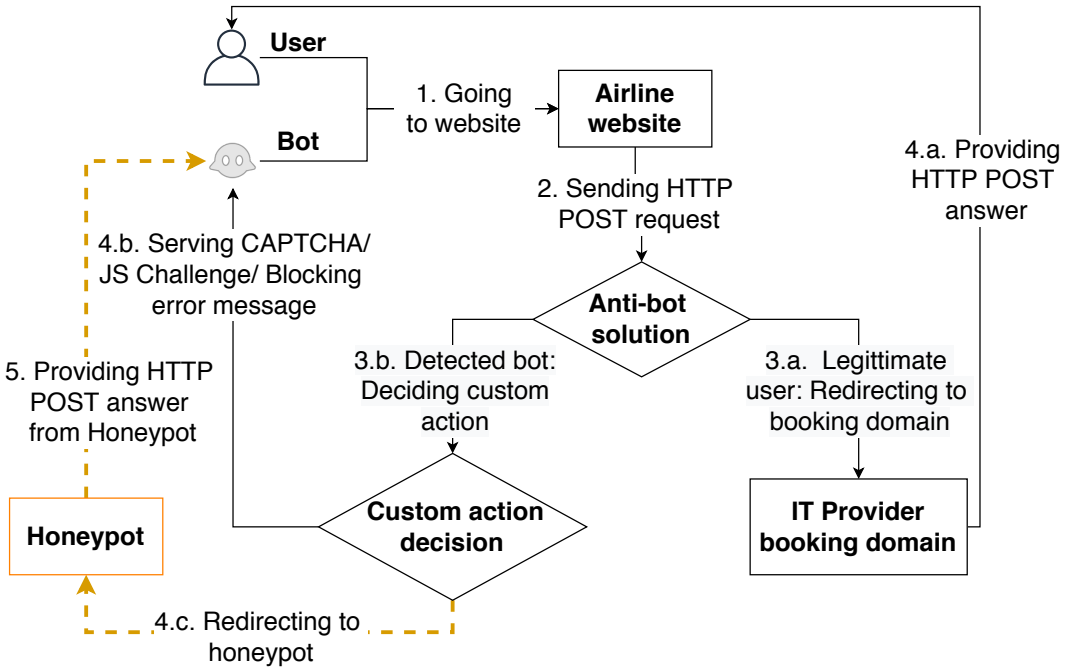

Figure 1. Scheme of the booking system. The dashed line shows the new addition of the honeypot into the flow.

4. System overview

The case studies were conducted in collaboration with a major IT provider specialised in airlines' websites. The provider offers different products for the booking process. For our case studies, we considered a specific product which we will refer to as Product A. When an airline company chooses Product A for its booking domain, they build a separate and private website, which is the point of interaction with the user. The user can indeed insert the details of his search (departure and destination locations, departure and return dates,...) in this domain. Once the process is finalised, an HTTP POST request with all the information is built and it is sent to the IT provider dedicated booking domain. As shown in Figure 1, the requests do not go directly to the booking website but pass through an anti-bot commercial solution. This artefact is used to recognise bot traffic through browser fingerprinting and mitigate it. If the request is detected as coming from a real user, the anti-bot solution redirects it to the booking domain. There, the value of the fare is computed in real-time, taking into account a huge number of variables, such as seasonal promotions and the number of seats left. This process is computationally expensive. Finally, the user receives back a web page containing the requested information.

When the request is detected as coming from a bot, custom actions can be put in place, like serving a CAPTCHA (Von Ahn et al., 2003), a JavaScript challenge, or blocking the bot.

In our setup, we implemented a new type of custom action, which consists of forwarding such requests to a honeypot, as shown with a dash-line in Figure 1. This platform, external to the IT provider environment, is capable of providing answers to requests while modifying the prices at will. The structure of the page to be served is obtained periodically from the real booking domain. Fares are retrieved thanks to an API of the IT provider. The system can modify the fares, insert them into the structure and send the response back. In this way, bots receive a response with the same syntax but different prices with respect to the original one.

For this work, we carried out a collaboration with a specific airline company, which uses Product A and will be called Airline A in this work. In general, 1 million requests are sent daily to that airline's website. The anti-bot solution usually blocks close to 40% of them. For each blocked request, there is an associated bot signature. The bot signature consists of a series of parameters that enables the anti-bot solution to link together requests coming from different IP addresses originating from the same source.

The anti-bot was configured in such a way that all the requests matching specific signatures, or a percentage defined by us, would be forwarded to our honeypot.

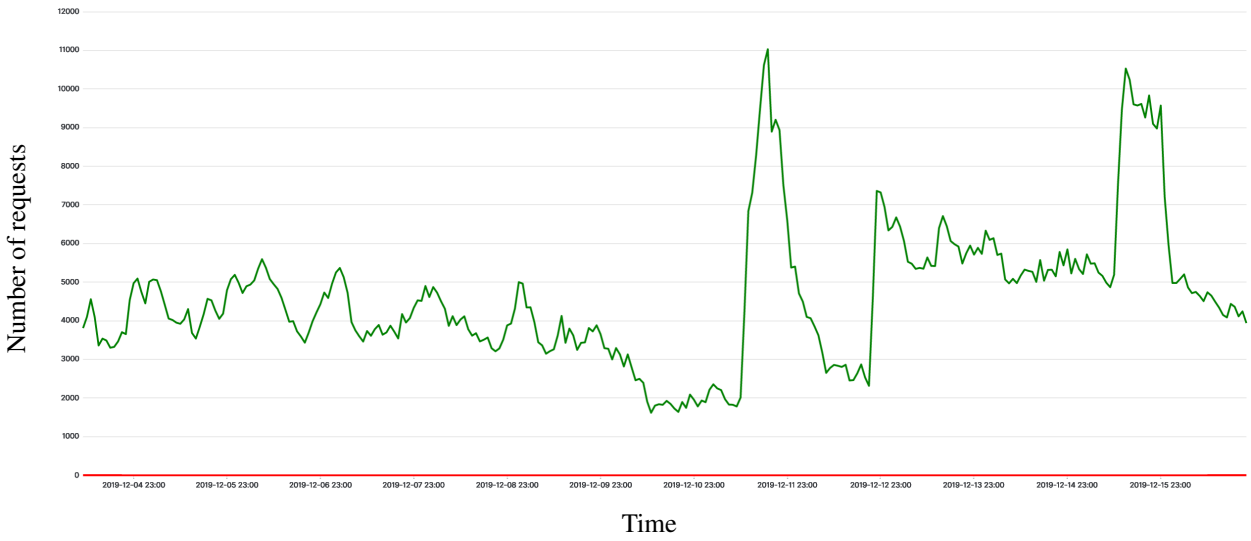

Figure 2. Amount of requests flagged with the chosen bot signature for the first case study and directed towards the booking domains of Airline A from 04/12/19 to 16/12/19 (before the beginning of the case study).

5. First case study

5.1 Method

A first case study was designed to test if the here above described system was able to lure bots. Our approach consisted of redirecting the traffic of a particular bot signature to the honeypot. We planned to serve the bots with correct answers for the first three days of the case study. The objective of this phase was to test that our modified setup was not creating particular artefacts that could be detected by the bots.

After this testing phase, the plan involved increasing the ticket fares by 5% for a random selection of 10% requests. The goal was to verify if the attackers were paying close attention to the results returned to them. Indeed, a careful observer, who was sending a large number of queries, could have noticed inconsistent results.

Moreover, while testing the possibility of luring attackers, we took advantage of the collected information to gather insights on the bots ecosystem. The collected HTTP payloads and IP addresses were used to answer the following questions:

- Is it possible to recognise a bot campaign from the information included in the payloads?
- Are bots crafting payloads to detect the honeypot?
- Can we derive meaningful information studying the patterns of bot IPs?

More in detail, the first question aimed to find out if behavioural patterns could be found in the HTTP payloads of the requests, something the anti-bot commercial solution does not look at. Indeed, bot signatures are built only based on request headers and browser fingerprinting. The payload of the HTTP requests contains different fields specifying the details of the searched flight, such as departure and destination airport. We decided to examine the distribution of all these parameters to possibly disclose particular artefacts specific to bots' payloads.

Regarding the second question, we wanted to understand if bots performed strategies to detect the honeypot. For example, they could have crafted particular payloads to see if the honeypot was answering differently from the original system.

To conduct our case study, we first identified the bot signature to be redirected to the honeypot by the anti-bot solution. We chose to look for it among the requests that were flagged by the anti-bot solution but which were not receiving any countermeasure (CAPTCHA, JavaScript challenge, Blocking error message) yet. We decided to take this approach because we did not want the bots to experience a sudden change in the type of responses they were getting. This could have biased the case study and the results.

The chosen bot signature was associated with traffic occurring throughout the whole day. Such behaviour had our preference, initially, as opposed to peaks of activity only in some moments of the day. We believed this was the best way to check if the bot was detecting any difference in the answers. Figure 2 illustrates the bot traffic in the days before the beginning of our case study. Its behaviour is not constant during the different days, but requests are sent continuously.

Because of technical limitations and a large number of requests flagged with the bot signature, we could not send to the honeypot the whole bot traffic. We decided to redirect there only 10% of the total traffic with the bot signature, while the rest would receive the original content.

We set as success criteria that, for 14 days after the application of the modified fares, the volume of traffic linked to the bot signature had to remain similar to the values registered before the beginning of the case study. To us, that would have meant that we were able to lure the attackers while serving incorrect information.

5.2 Results

Figure 3 displays all the changes in the traffic during the case study. The x-axis shows the running time of the case study (between 9:00 GMT of the 16th of December 2019 and 18:00 GMT of the 18th of December 2019). On the y-axis, the number of received requests is displayed. Coloured squared are used to emphasize the different phases of the attempt.

Initially, (blue square) the case study was conducted without any problem. We were planning to apply the changing prices strategy, as scheduled, 72 hours after the beginning of the attempt. However, after only 23 hours of running time (red square), we observed a drastic drop in all the traffic corresponding to the bot signature. This traffic included both the requests redirected to the honeypot and the ones arriving at the real booking system. The traffic remained then stable during the next 24 hours (violet square).

After these 47 hours from the beginning of the attempt, we decided to increase the percentage of traffic redirected to the honeypot. We wanted to understand if the bots were reacting to the honeypot presence or if this change was only due to the mutable nature of the bot behaviour. Indeed, as shown in Figure 2, the traffic associated with the bot signature was not constant on different days. Instead of 10%, we started redirecting 50% of that bot traffic to the honeypot, leaving the other 50% receiving the original content (orange square). Thirty minutes after the implementation of this decision, we experienced a disappearance of the traffic associated with the bot signature.

In total, the case study ran for 58 hours. We had to stop it due to the complete drop of traffic volume of the bot signature, which happened after around 48 hours from the beginning of the case study. Thus, we could not test the modification of the prices. In that regard, our case study was a failure. On the other hand, as explained here below, we learned something new from the *post-mortem* analysis of the logs.

We examined the collected requests to understand why we had failed. The distribution of the COMMERCIAL_FARE_FAMILY (later referred to as CFF) parameter picked our attention. Given that Airport A is the name of the base airport of Airline A, there are three possible values for the parameter:

- **VAL1**: for fares from/to Airport A to/from North American locations,
- **VAL2**: for fares from/to Airport A to/from European locations,
- **VAL3**: for fares that do not have Airport A neither as departure nor as arrival locations.

Table 1 shows, for each possible value of the CFF parameter, the number of occurrences and the percentage of this value with respect to the total number of requests in the case study (12,801). Since this data shows that the value VAL3 appears only in 0.03% of the requests, we expected to have Airport A as departure or arrival locations in almost all the requests. However, as displayed in Figure 4, there are multiple combinations of departures and arrival locations that do not include it. In the picture, the departure (the inner circle) and the arrival (circle in the middle) locations, as declared in the parameters, are shown. The bright red splices represent Airport A, whereas other colours are used for the remaining locations.

To better understand this situation, we inserted also the CFF value in the same chart (most exterior circle). Even if VAL3 is almost not shown, Airport A is not the departure or destination location of all the flights. When Airport A is not present the value of the parameter is equal to VAL2, which is incorrect since it violates the semantic of that field.

Given the high number of parameters in the HTTP POST payload (one hundred and four), we did not perform a detailed study of the combination of these parameters before the beginning of the case study. In particular, we had never examined the correlation between CFF and the success of the request. We tried to query the real domain inserting a wrong value of this parameter and we discovered that a “Non Availability Page” was sent back, even if fares for those requests would be available. In the honeypot, the same requests would return the correct answers, since we did not cover this case.

We made the hypothesis that the bots used this case to leak information about the validity of the responses the system was providing them. To check this theory, we studied all the requests coming to the honeypot which would have received a “Non Availability Page” in the real system, but got a correct result in our platform. Thus, we examined the requests in which CFF was equal to VAL2 and neither departure nor arrival locations were Airport A.

Our objective was to understand if this type of request was produced for the whole time of the case study or from a defined moment on. Hence, we created a plot that we interpreted as a “scatter plot” of the reception of these particular requests over time.

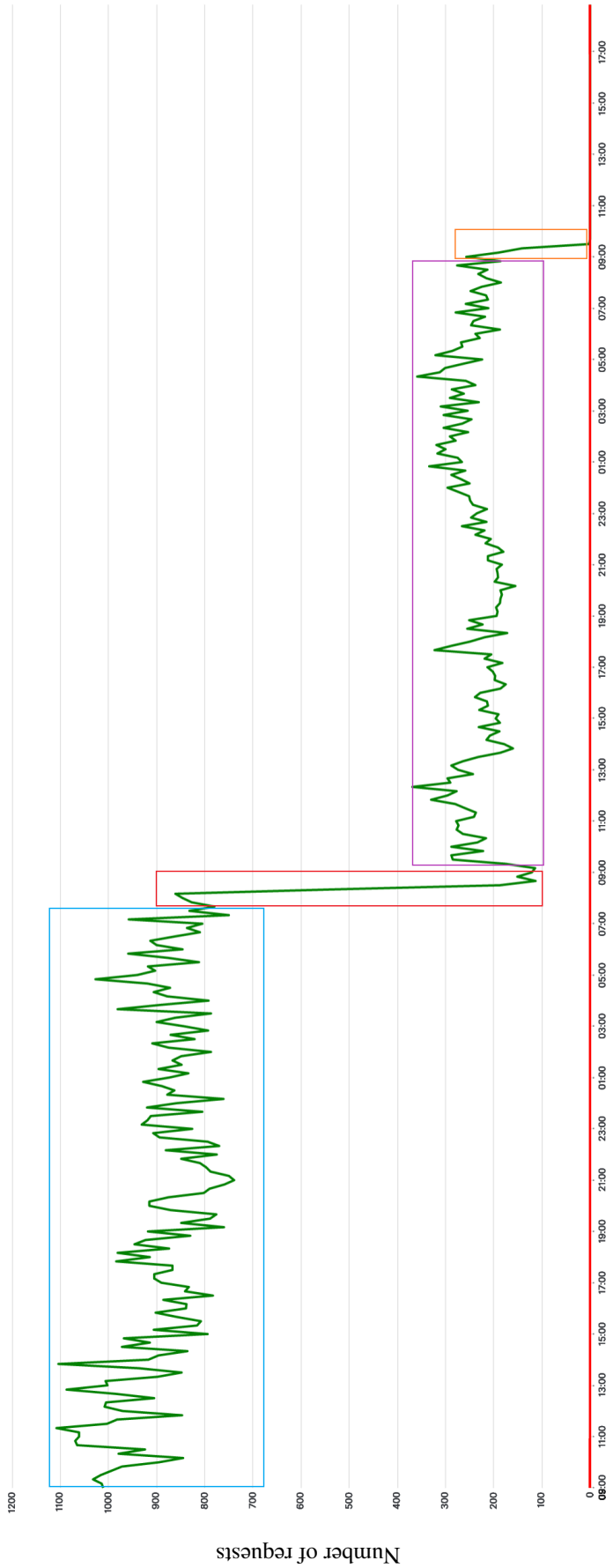

Figure 3. Amount of requests reaching the honeypot during the running time of the first case study (from 16/12/19 9:00 GMT to 18/12/19 18:00 GMT). Coloured squares highlight the different phases of the case study.

We plotted the semantically wrong requests on the timeline of the case study, as shown in Figure 5. On the x-axis, we can see the timestamps of the running time of the case study. Each row inside the chart corresponds to an arrival airport. Each column of each row counts the number of requests per hour with the combination of the arrival airport of the row and one or more departure airports, defined with different colours. For example, the fifth row from the top of the plot (highlighted in a red square) represents all the requests for flights arriving in Airport Y. These requests have been divided with respect to the corresponding hour of their timestamp: if any flight for Airport Y was requested in an hour, there is a corresponding column with the count of requests for that hour. Columns coloured in blue show the flights departing in Airport X and arriving in Airport Y. Columns coloured in violet show the flights departing in Airport Z and arriving in Airport Y. When requests with both departing airports were made in the same hour, columns of different colours are one on the top of the other.

Looking at the plot, we can notice that the requests were sent constantly from the beginning of the case study to the final drop of the traffic.

To understand if this behaviour was particular to the case study or this type of requests was produced also before, we would have needed to study the requests created by the bot signature before the beginning of the case study. Unfortunately, the payload of the requests was not normally collected. However, a short test of around three hours had been conducted in the days before the case study on the same bot signature. The goal of this test was to verify the correctness of the system. We used the data collected during this test to perform our analysis.

A plot similar to the one produced for the requests of the case study was built with the data of the test, as illustrated in Figure 6. The semantically wrong requests were collected and plotted on the timeline of the test. The x-axis represents the timestamp of the requests and each row corresponds to an arrival airport. Every column expresses the number of requests received every five minutes with the combination of the arrival airport of the row and departure airport(s). Different colours represent different departure locations.

The picture displays how the particular requests were not initially sent and their reception only started after around 2 hours from the beginning of the test.

Constructing this kind of requests is peculiar because, since they are not built in a semantically correct way, they do not lead to a successful reply in the real system. They represent an economical loss for the bots, which invest resources not to get the information they are looking for in return. This fact, correlated to the apparition of this kind of requests only after the beginning of the tests with the honeypot, leads us to the theory that the bots could have used this technique to spot changes in the environment.

Once we found out this possible detection technique, we modified the implementation of the honeypot to cover this corner case. Before proceeding with other case studies, we decided first to check if there was value in the information we already collected. We analysed the 12,801 HTTP POST requests collected during the attempt and we found out distinctive cases.

The TRIP_TYPE parameter depends on the type of trip: return, one way, or multi-city (round trip in which the departure of the first flight does not coincide with the destination of the second one). Studying the logs we noted that all the requests which arrived at the honeypot had this value set for one-way flights. Since return flight prices cannot be built directly from the ones of the single fares, this behaviour appears peculiar because bots were losing possibilities to have a complete view about Airline's A flights. A possible explanation could be that this bot signature was just one of the many ones used by the botnet and there was a distribution of the workload. On the other hand, it could also be that the business of the botnet behind this attack was only based on one-way flights.

A validation of these theories can be found in the parameters which express the language and the country of the requester, as declared in the request. Both values were set to US in all the examined requests. There are sixteen available combinations of these parameters for Airline A, therefore this homogeneity shows as well a selective behaviour of the bots.

This information first gives us confidence that all the requests were issued by the same actor, since there are behavioural similarities. Moreover, it led us to think that detection based on the behavioural patterns should be further investigated.

The last question we proposed was about collecting a set of IPs in which to search for patterns. Since the data set was small we did not consider it meaningful to proceed with further analysis on it.

6. Second case study

6.1 Method

The first case study showed us it was possible to acquire insights about the botnet ecosystem. Thus we decided to perform another attempt, after modifying the implementation to cover the corner case previously discovered. This second case study is also reported in our previous work (Chiapponi, Catakoglu, Thonnard, & Dacier, 2020).

The goal of the case study was the same as the one of the first attempt: testing the possibility to lure the scrapers, randomly modifying the fares of 10% of the requests by 5%. Moreover, we wanted to find answers to our questions regarding the possibility of recognising bot campaigns from the payloads, understanding if the bots were detecting the honeypot, and deriving patterns from the bots IP addresses.

Value	*Count*	*Percentage*
VAL1	6,401	50.00%
VAL2	6,396	49.97%
VAL3	4	0.03%

Table 1. Number of occurrences and the percentage of the CFF parameter in the requests of the first case study.

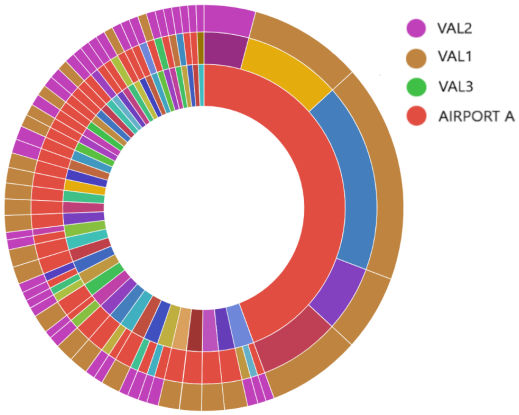

Figure 4. Pie chart representing the departure (inner circle) and the arrival (median circle) locations and CFF parameter (outer circle) of the requests of the first case study.

Examining the previous case study, we decided to make some changes in the setup, in particular regarding the traffic pattern of the redirected bots. The goal of the changes was to eliminate every possibility of uncertainty, to be able to clearly define if, in case of failure, this was caused by the detection of the honeypot.

Since the behaviour of the bot signature of the first case study was not constant on different days, we could not arrive at certain conclusions regarding the drop in the traffic. Thus, we made a requirement that the activity of the bots for the new case study was regular during different days.

Furthermore, we considered that we could have introduced a bias in the first attempt, because only a percentage of bot traffic was redirected to the honeypot. Thus, we decided to choose a bot signature whose traffic, even with our technical limitations, could be entirely diverted to the honeypot. In this way, we could have better determined if the honeypot had been recognised. Indeed, if the bot activity had dropped drastically or stopped completely, it would have meant that our approach had failed.

In order to respect these requirements, among the bot signatures which were not receiving any countermeasures, we chose a signature with peaks of traffic only in particular moments of the day. The plot of transactions over time for the selected signature is shown in Figure 7. The bot traffic presented daily peaks of activity, with varying amounts of traffic in the range of 300 to 500 daily requests, all received during the same 40 minutes time window. Every day the peak repeated itself in the same period of the day.

We planned, as for the previous attempt, to serve the original fares in the first 72 hours and to start luring the attackers after that. We would have considered the case study successful if, for at least 14 days after the beginning of the luring phase, we would have seen daily peaks of activity as in the days before the attempt.

6.2 Results

The case study ran between the 7th of January 2020 and the 2nd of March 2020, for a total of 56 days. After 72 hours from the beginning of the attempt, we started modifying the fares, increasing by 5% a random selection of 10% requests.

The matching requests drastically disappeared all on March 3rd. We believe the reason is linked with the business needs of the actor behind these bots. Indeed, that date coincides with the beginning of the worldwide COVID-19 pandemic. Furthermore, the airline, subject of our case study, is the main one for a country whose government issued its first major travel restriction on the 2nd of March, practically shutting down airline travel to and from that country. Without any customer interested in buying tickets to/from that country, there was no incentive for the malicious actor to keep collecting ticket prices from that company. This and the fact that the bot behaviour did not change after the increase of the price most likely explains the disappearance of these bots.

In total, the platform served modified fares to the bots for 53 days. Thus, according to our set of rules for defining success, the outcome of the study was considered favourable.

Over the duration of the case study, the honeypot received 22,991 requests. The daily average amount was 410 with a standard deviation of 33 queries. All requests arrived at the same time of the day. The signatures were only seen during a small time window, averaging 38.18 minutes. The amount and the timing of the requests were in line with those of that bot signature before the beginning of the case study, even after the beginning of the modifications.

Changing randomly some of the values without causing the bots departure, we learned that i) they do not have a ground

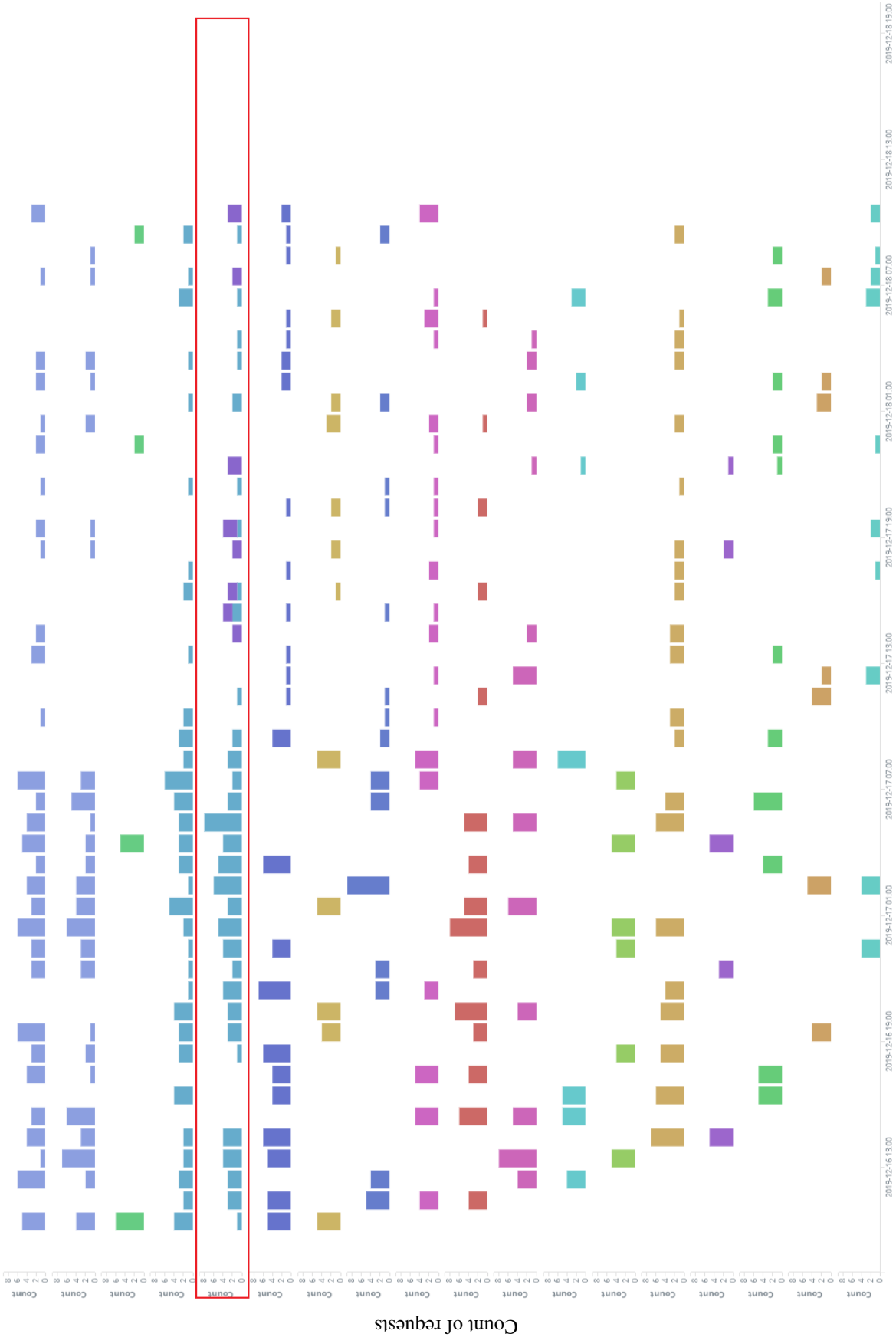

Figure 5. Requests of the first case study with combinations of departure and destination airports that do not contain Airport A and have the parameter CFF set to VAL2.

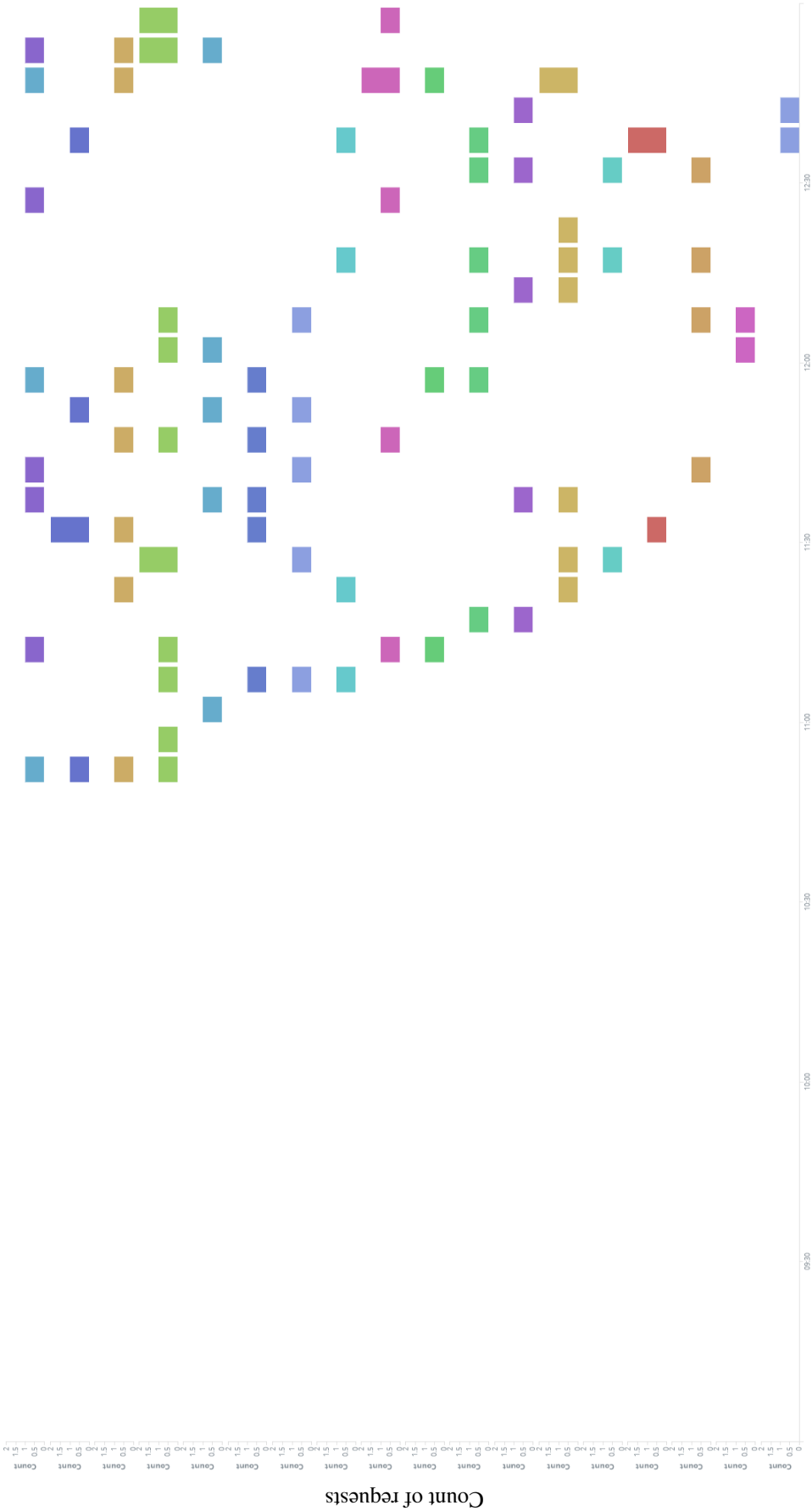

Figure 6. Requests of the 3 hours test with combinations of departure and destination airports that do not contain Airport A and have the parameter CFF set to VAL2.

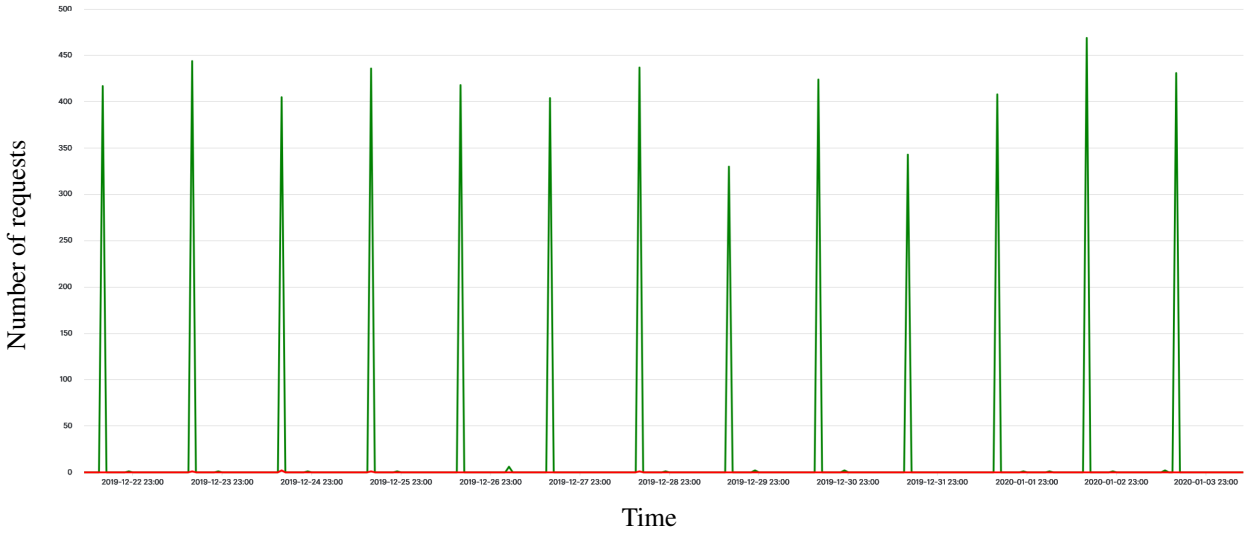

Figure 7. Requests flagged with the chosen bot signature for the second case study and directed towards the booking domains of Airline A from 23/12/19 to 03/01/20 (before the beginning of the case study).

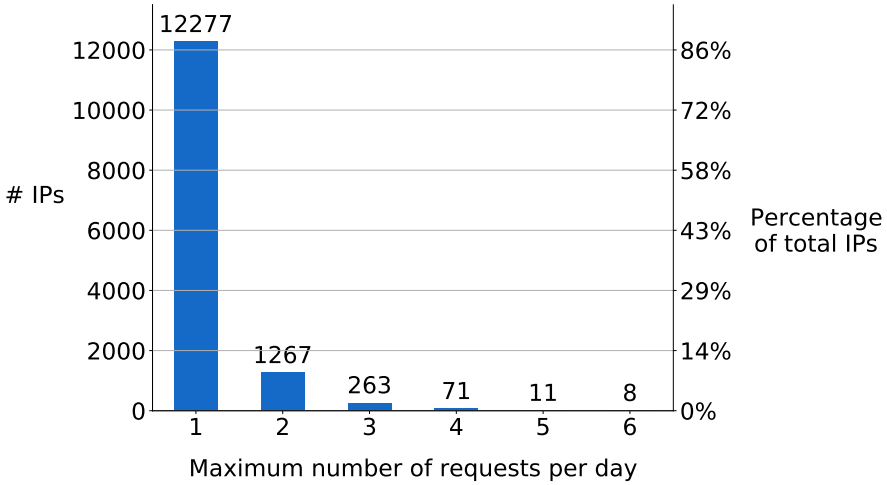

Figure 8. Left (resp. right) Y axis: absolute (resp. relative) amount of IP addresses which have made at most X requests per day

truth to compare the returned values and/or ii) their plausibility check is not sophisticated enough to detect small changes, not even by correlating values collected over several days.

The 22,991 requests were issued by 13,897 unique IPs. These IPs belong to 1,187 /16 blocks which means that, on average, there were less than 12 IPs (mis)used by bots within each /16 block, *i.e.* less than 0.02% of that IP space. Furthermore, geo-localisation of these IPs indicates 790 distinct origins, in 86 different countries. This highlights how widespread the misused IPs are on the Internet.

However, in the pool of collected IPs, we saw different repetitions, which led us to study them in-depth. Figure 8 shows that most of the IPs (97% of the total) made at most two requests per day, with the vast majority (88%) making only one request per day. Figure 9 shows the total amount of requests made per distinct IP over the whole case study. Here, we see that 8,257 IPs have sent only one request. This value is to be compared with 12,277 of Figure 8. It highlights the fact that a large amount of IPs have shown up on at least two different days, issuing a single request every time. This is confirmed by Figure 10 where we see that almost 30% of the IPs have been seen on at least two different days.

After studying the IP addresses, we examined the HTTP payloads of the requests. First of all, we checked if the bots were producing semantically incorrect queries, as in the first attempt, to possibly detect the honeypot. We did not find any evidence of this. Hence, we made the conjecture that the redirected bot signature of that attempt corresponded to a different botnet.

Inspecting the payload, we found a striking similarity among them. This is consistent with the idea that they all are issued by bots obeying to the same operator for a repetitive data collection task.

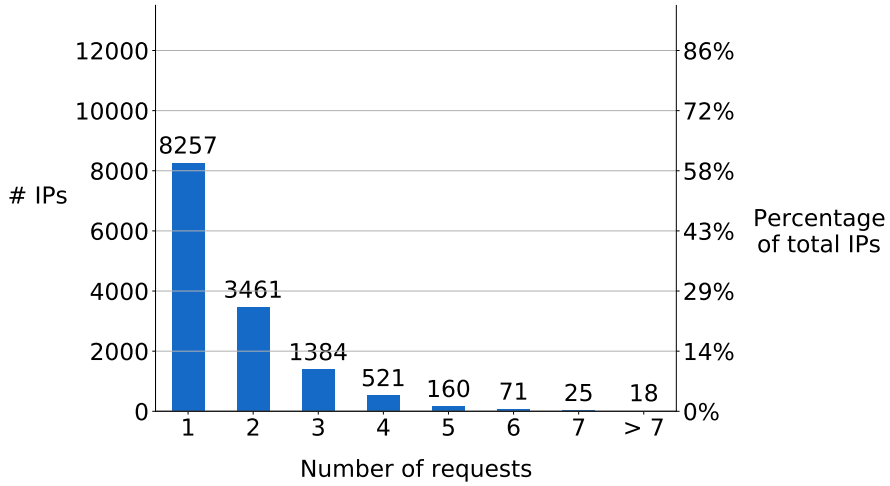

Figure 9. Left (resp. right) Y axis: absolute (resp. relative) amount of IP addresses which have made at most a grand total of X requests during the whole period of the case study.

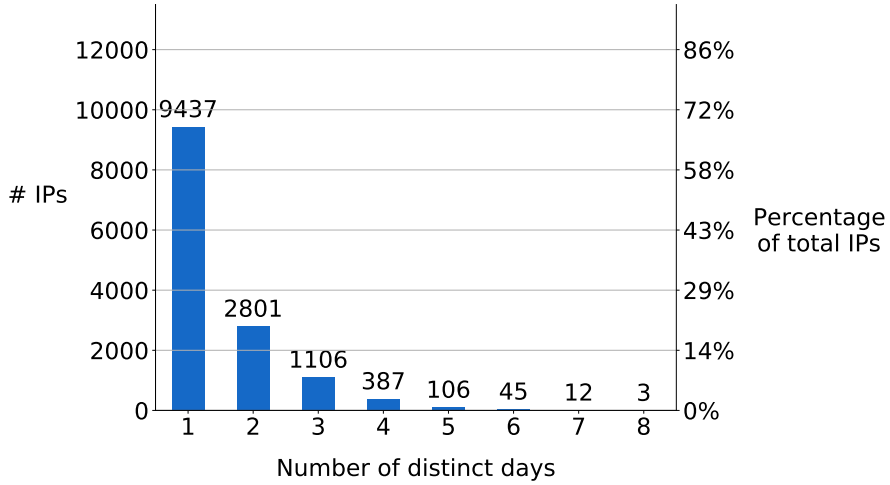

Figure 10. Left (resp. right) Y axis: absolute (resp. relative) amount of IP addresses which were seen in X distinct days during the whole case study.

The requests were interested in return flight tickets (resp. one way) in 51.5% of the cases (resp. 48.5%) where the return date was always 7 days after the departure one. The 22,991 requests only look for 25 combinations of 16 departure and 12 arrival airports from where this airline operates flights, an extremely small fraction of the airline offers.

The time interval between the date of the request and the one of the departure was also pretty regular. It was either between 0 and 14 days or 21, 30, 45, 60, 90, 120 days. Only a few requests (48, 0.2% of the total) had different values (20, 31, 44 days) but they were also done outside the 40 minutes window and probably do correspond to a different phenomenon. In Figure 11, we represent the distribution of these time intervals among different types of segments (combinations of one-way/return flight, departure, and arrival locations). For each segment and specific interval, we calculated the number of observed requests over the 56 days of the case study. For example, the data corresponding to the value 0 on the x-axis, represents the count of all the requests which asked for a flight departing the same day in which the request was made, thus with an interval equal to 0 days. The boxplot is built by first counting, for each combination, how many requests have this interval. Then median and percentile values are calculated for this aggregated data. The same process has been repeated for intervals of all sizes. These results clearly show that the various intervals have similar distributions of values.

Figure 12 looks at this regularity from a different angle. Now, for each request date, we compute the amount of observed time intervals for all paths taken together. Similarly to the previous figure, the data corresponding to the value 0 on the x-axis represents an interval equal to 0 days. The boxplot is built by counting the occurrence of this value in the requests, grouping them according to the request date. The new boxplots show the same regularity in the querying process: no matter how we look, we see the same kind and amount of queries being done, day after day. They are not identical though. If we compute the

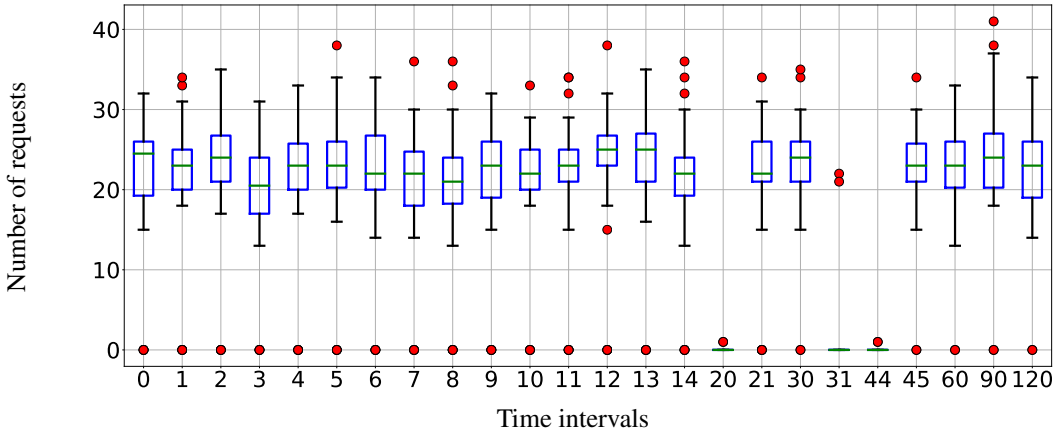

Figure 11. Boxplots representing the distribution of the intervals among different paths.

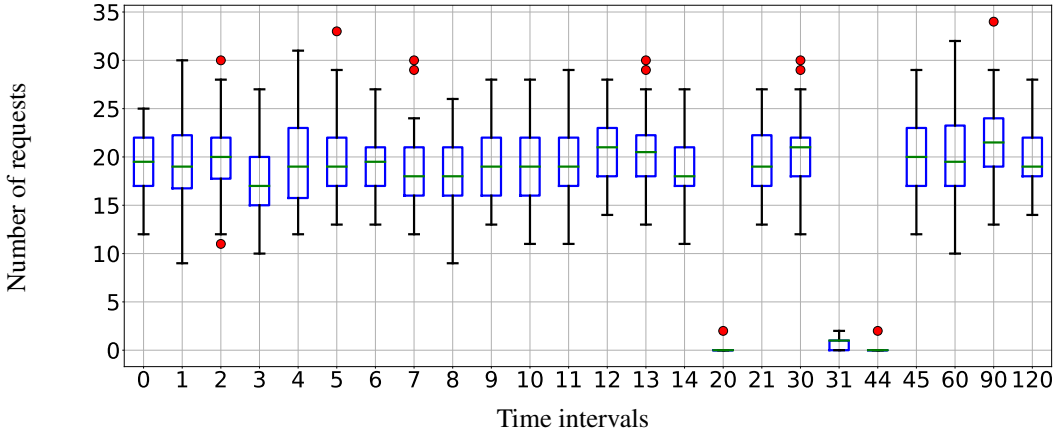

Figure 12. Boxplots representing the distribution of the intervals among different request dates.

4-tuples made of i) departure airport, ii) arrival airport, iii) time interval, iv) type of flight (one way or return), we can identify 982 distinct ones over the whole case study, to be compared with the average 410 requests per day.

Studying the occurrences of the different tuples, we found that on average each tuple is asked 23.41 times during the whole running time of the case study. This value is pretty close to the average number of days in which a tuple is requested, which is 22.85. This shows how generally each tuple is asked just once a day at most. However, 20% of the tuples have been asked, at least on one occasion, more than once a day. The maximum number of times the same tuple has been requested on the same day is 8. Almost each day (every day but one) at least one tuple has been queried more than once. Requests asking for the same tuple are always asked in a little period of time. The maximum distance between two of them is 337 seconds (around 5 minutes and a half), the minimum is 5 seconds while the average value is 45.2 seconds.

This behaviour is somehow peculiar because not only the same combinations are asked multiple times, but this is done in a short amount of time. One possible explanation is that a check of consistency for the prices is put in place. However, as explained before, the bots did not find discrepancies in the prices even after the 5% increase was put in place. This would mean that their threshold for an anomalous situation is higher than such value. Another possible explanation could be that a bot did not return an expected reply quickly enough, causing its master to ask another bot to submit the same request a second time.

Finally, one of the questions we wanted to answer regarded finding meaningful patterns among the bots IP addresses associated with a specific bot signature. In this second case study, we considered the data set appropriate for further analysis and we proceeded with them, as shown in the next sections.

Booking Time	Request Time
2020-01-17	2020-02-01 2020-02-05 2020-02-14
2020-02-26	2020-01-10 2020-01-23
2020-02-29	2020-02-01
2020-02-06	2020-02-23
2020-02-07	2020-01-24 2020-02-02 2020-02-19

Table 3. Timestamp of the bookings and the honeypot requests made by the same IPs.

Score (S)	% of IPs	# of IPs
$S < 75$	28 %	3958
$S \in [75, 85[$	46 %	6371
$S \geqslant 85$	26 %	3568

Table 2. Distribution of the fraud score (IPQualityScore)

Type	Number of Ips	Percentage
VPN	180	0.013
Proxy	59	0.004
Hosting	1733	0.125

Table 4. IPInfo.io classification of the IPs

7. IPs reputation

The IP addresses collected during our second case were studied to check if the bots were taking advantage of proxy services, as discussed in our previous work (Chiapponi, Dacier, Todisco, Catakoglu, & Thonnard, 2020).

The collected IPs are supposed to be residential IPs; i.e., they belong to legit users who could, possibly, be interested in buying tickets. To verify this, we have looked for the presence of these IPs in the logs of 17 other airlines. We found out that during the case study, five bookings have been realised by 5 of these IPs. In Table 3 we indicate when the booking was done vs. when the same IP was seen in our honeypot logs. As expected, the dates differ greatly. Moreover, none of these requests had the bot signature associated with them. They look perfectly legit. This confirms two things:

1. Some of these IPs are likely used by legit users,

2. The risk of blocking legit customers when blocking identified proxy IPs remains extremely small.

On the other hand, the simplest way to implement a proxy is to open some ports and have a proxy server listening behind them. This should thus be detectable by the various actors who scan the Internet continuously, looking for threats and/or vulnerabilities. We have used two such systems to see if they had identified our IPs as behaving like proxies. First, we have used IPInfo.io (*Comprehensive IP address data, IP geolocation API and database - IPinfo.io*, n.d.) which provides a boolean value for each IP in the categories "VPN", "Proxy", "Hosting". According to the provider of that service, VPNs are usually encrypted traffic endpoints so typically, if there is a VPN operating on an IP address, there will be either encrypted traffic or ports open which will obviously show a VPN is being used. Proxies are usually just a "HTTP forwarding" service and redirect traffic to somewhere else (internal domains, other servers, etc) (IpInfo.io, August 2020)."Hosting" category specifies if the IP belongs to hosting providers.

Table 4 shows that a couple of IPs have been categorised as involved in suspicious activities but not as many as expected. However, the results obtained with IPQualityScore (*Fraud Prevention | Detect Fraud | Fraud Protection | Prevent Fraud with IPQS*, n.d.) are much more aligned with our expectations. As explained in their documentation, this service tells if an IP has been used in "automatic fraudulent behavior" in the "Bot status" category, while indicating a positive value of "Recent Abuse" if the IP has been involved in a "recently verified abuse across their network". The abuse behaviour includes charging back, compromised devices, fake app installation. Moreover, the "VPN" category indicates server or data center IPs that allow tunneling. Finally, the "Proxy"[2] category, identifies a device being infected by malware, a user selling bandwidth from their connection, or other types of proxy like SOCKS, Elite, Anonymous, Tor, etc. With this service, we can notice that the number of IPs involved in malicious activity is much higher in comparison to the first one. Furthermore, this service provides a general fraud score for the IP: this value ranges from 0 to 100, indicating a suspicious activity when higher than 75 and high risk when greater than 85. Table 2 tells that around 72% of the IPs show suspicious behaviour, of which 26% are classified as high risk. This is quite consistent with the idea that malicious actors are hiding behind them, ruining the reputation of these IPS.

To dig deeper into the analysis of the malicious behaviours associated with these IPs, we looked for their presence in anti-spam DNS blocklists. Using the Python library Pydnsbl we checked multiple blocklists and we found out that 76% of the

[2] A "VPN" is automatically a "Proxy" according to their definitions

Type	Number of Ips	Percentage
VPN	9138	0.658
Proxy*	1075	0.077
Recent abuse	3878	0.279
Bot Status	2780	0.200

Table 5. IPQualityScore classification of the IPs (*From the total number of positive matches, 10213, we subtracted the number of positive values of VPN)

Date	Number of Tor Ips
2020-02-14	12
2020-02-15	24
2020-02-18	20
2020-02-19	40
2020-02-24	12

Table 6. Number of tor exit nodes which had an IP found in the honeypot on the corresponding date.

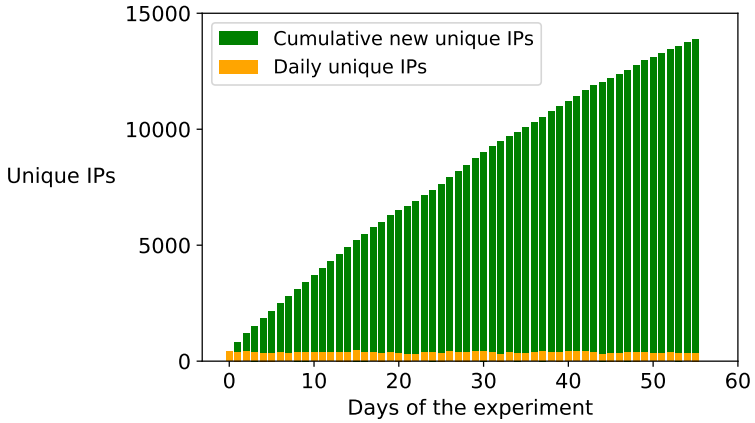

Figure 13. Cumulative curve of the new unique IPs in comparison with the daily unique IPs.

IPs were blocked at least in one of them at the time of our analysis (July 2020). Hence, we had the confirmation that these IPs were doing malicious activity also outside of our environment.

Finally, to better understand if the IPs collected in the honeypot were involved in other hostile activities, we checked if they were used in the Tor network (*Tor Project*, n.d.) during the time of the case study. We analysed day by day if the IPs seen in the honeypot were advertised as exit nodes. We discovered that 72 IPs were announced in this way on 5 different days. In Table 6 we provide the details about the dates and the number of IPs found each day. It is peculiar how only 5 days close in time have witnessed this match. We hypothesise that the bots tried to take advantage of the tor nodes but they did not continue with this strategy.

All the analyses show how the collected IP addresses were used by various actors in different and possibly malicious contexts. This gives us confidence in affirming that the IPs belong to proxy services and the actor behind the botnet is taking advantage of this service.

8. Modelling and Proxy Services

8.1 Introduction

As discussed in Section 6, 30% of the IPs collected in the second case study have been seen on at least two different days. This number is surprisingly high. Indeed, at this stage, we have to remind the reader that these IPs are, most likely, proxy IPs and that the actual client machine sending a request is hidden behind. The proxy service offers a pool of addresses to be given to these clients. Let us call B the size of the pool assigned to a botnet. Figure 10 shows how many times a given address has been picked over a period of 56 days by the botnet of our second case study. The fact that 2,801 IPs have been used twice over that period is inconsistent with the assumption that the addresses would be randomly picked out of a very large pool of millions of IPs. Indeed, to calculate the probability that a given IP got picked twice over this period comes down to resolving the classical birthday paradox which can be generalised as follows:

> *Given n random integers drawn from a discrete uniform distribution with range [1,d], what is the probability p(n; d) that at least two numbers are the same? (d = 365 gives the usual birthday problem.)* (Suzuki, Tonien, Kurosawa, & Toyota, 2006)

In our case, n is equal to 56, the number of days where IPs from the pool are assigned to the botnet and d is equal to the size

of the pool B. We want to assess the probability that the same IP would be drawn twice over that period of 56 days. We can rephrase the birthday problem for our needs as follows:

> *Given 56 random integers drawn from a discrete uniform distribution with range [1,B], what is the probability* $p(56;B)$ *that at least two numbers are the same?*

The formula $1-(\frac{B-1}{B})^{\frac{56(55-1)}{2}}$ gives an approximate result:

- If $B = 10000000$ then $p(56, 10M) \approx 0.000154$
- If $B = 1000000$ then $p(56, 1M) \approx 0.001538$
- If $B = 100000$ then $p(56, 100K) \approx 0.015282$

Clearly, considering that we have seen more than 30% of the IPs drawn at least twice, either B is significantly lower than the number announced by the proxy services, or the assignment of IPs is not randomly done, or both.

Regarding the total amount of IPs, we saw only 13,897 different ones. Every day the number of distinct IPs, on average 371 (shown in yellow in Figure 13), was similar to the number of requests, on average 410. Thus, it is clear that most IPs send a single request on a given day and do the same thing again a few days later. In the same figure, the green columns represent the cumulative number of unique IPs observed in our honeypot since the beginning of the case study. The figure shows that the daily increment decreases over time, suggesting that it will eventually reach a maximum.

To better characterise and understand the threats ecosystem we are facing, we tried to find a mathematical model that approximates as closely as possible the assignment of IPs made by the proxy provider. We used that model to derive the most likely size of B.

Leveraging our previous work (Chiapponi et al., 2020), we propose hereafter two distinct modelling approaches to assess the most likely size of the pool of IPs B put at the disposal of the observed bots. Both models deliver a value that is below 70K, i.e. orders of magnitude less than the tens of millions of IPs supposedly provided by proxy services.

In the first approach (subsection 8.2), we look at the IPs assigned every day by the proxy to the bots. We model this as a drawing process with different probability distributions and we try different pool sizes $\mathscr{P}$ to find the one that shares the same characteristics with the ones we have witnessed. We consider the best $\mathscr{P}$ we find as the value of the botnet pool B.

In the second approach (subsection 8.3), we look for a fitting curve to approximate the one shown in Figure 13 and, by extrapolating it, seeing what maximum value it would reach, and when.

8.2 IP assignment as a drawing process

8.2.1 General principle

Figure 10 tells us how many IPs have been assigned to a bot only once, or twice, or three times... over the duration of the case study. We model this assignment process by a daily probabilistic drawing process without replacement. We arbitrarily define a pool size $\mathscr{P}$. On a given day, we draw from our pool, without replacement, several values equal to the amount of distinct IPs seen that day. We do this every day, keeping track of which value got drawn several times during this exercise. We use these accumulated results to produce a histogram similar to Figure 10.

We use the Wasserstein[3] distance to assess the similarity between this histogram and the one from Figure 10, making the reasonable assumption that the values of the produced histogram are derived from the real one by small and non-uniform perturbations.

We have no reason to believe that the drawing is done every day instead of every 2 days or 3 days or more. We thus repeat the process with other window sizes s (2 to 10), but we proceed with a drawing with replacement. We impose an additional constraint though. A given value cannot be drawn more than s times, i.e., once per day. Once a value has been drawn s times, it is not replaced in the pool anymore.

We use different probability distribution functions to ensure that they do not produce drastically different sizes. Various other functions could have been used. Our goal is not to find the best one but to show that several "good enough" ones deliver the same ballpark figure for B.

[3]This distance is known as the earth mover's distance since it can be seen as the minimum amount of "work" required to transform one histogram into another, where "work" is measured as the amount of distribution weight that must be moved, multiplied by the distance it has to be moved (Levina & Bickel, 2001).

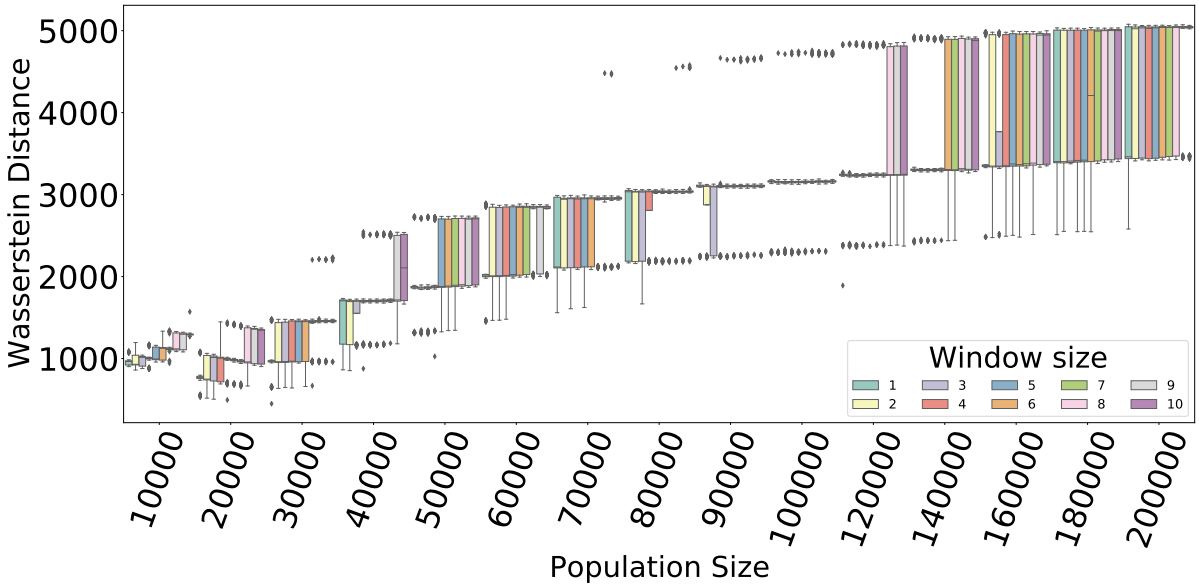

Figure 14. Uniform distribution: for each population size on the x axis, a group of boxplots displays the Wasserstein Distances (y axis) obtained in the 100 experiments for that population size. Each color represents the window size used for the simulation.

8.2.2 Used algorithm

We studied the distribution of the IPs for subgroups of days of size s ranging from 1 to 10. To group the days, we have used juxtaposed windows (as opposed to sliding windows) to ensure that our final histogram contained the same amount of values as the one in Figure 10. We chose juxtaposed windows to not count twice the IPs of a singular day and reproduce thus a coherent replica of the observed data.

We have run simulations with different population sizes $\mathscr{P}$. We have incremented $\mathscr{P}$ by 10,000, starting with the initial value of 10,000 up to 100,000. Moreover, we have tested values from 100,000 to 200,000 with an increment of 20,000. For a given time window, we have produced as many histograms as distinct population sizes. Each histogram is obtained thanks to 100 simulations. Each simulation produces its histogram and we compute the Wasserstein distance between this histogram and the empirical one. An average Wasserstein distance value is then obtained from these 100 simulations. The lowest value of this average distance corresponds to the size $\mathscr{P}$ which best represents the observed data. For each window size, for each population size, for each simulation, we have plotted the distances obtained using a boxplot representation. This algorithm has been applied using three distinct probability distribution functions, as explained here below.

Uniform Distribution The simplest model is the one where all IPs, every day, have the same probability of being assigned to a bot. To model this, we use a uniform distribution as the probability distribution function in our drawing process. Figure 14 shows for each window size (colored legend) and for each population size (X-axis), the boxplots of Wasserstein Distances (Y-axis) obtained in all the experiments. We clearly see that for $\mathscr{P}$ bigger than 30K, the bigger its value, the more different is the obtained histogram with respect to Figure 10. The best distances are obtained for the low value of $\mathscr{P}$ of 20K, for all time window sizes. The Wasserstein distance is quite high though, around 1,000 and we have looked for other distributions with the hope of obtaining smaller distances.

Gaussian Distribution (aka normal) It is reasonable to imagine the existence of a bias in the IP assignment process that would lead some IPs to be more frequently used whereas others would be rarely picked. This could be due, for instance, to the simple fact that some residential IPs might be more frequently available (online) than others. Another reason could be that proxies, to ensure a better quality of service, assign preferably IPs "close" to their customers. Our goal is not to identify the causes of these biases but, simply, to assume that they could exist and, thus, model this possibility. To do so, we have run our algorithm with a Gaussian distribution. For the sake of concision, the results presented here correspond to the parameters mu=0.5 and $sigma$=0.1. Other choices lead to the same lessons learned and this combination offers the best distances. We offer in Figure 15 a similar representation as in Figure 14. This model seems to be a better approximation since the best Wasserstein distance is now half of the one obtained for the uniform distribution. As expected, the size $\mathscr{P}$ does grow since several IPs are now very rarely chosen. Its value, around 60K, is still three orders of magnitude below the claimed 70M.

Beta Distribution Last but not least, we present also the results obtained with the Beta distribution, with $alpha$=1 and $beta$=5. This distribution enables us to represent a different form of bias in the choice. However, the results are very consistent with an optimal size P of 60K, as in the Gaussian distribution, and a Wasserstein distance below 500. The results are represented

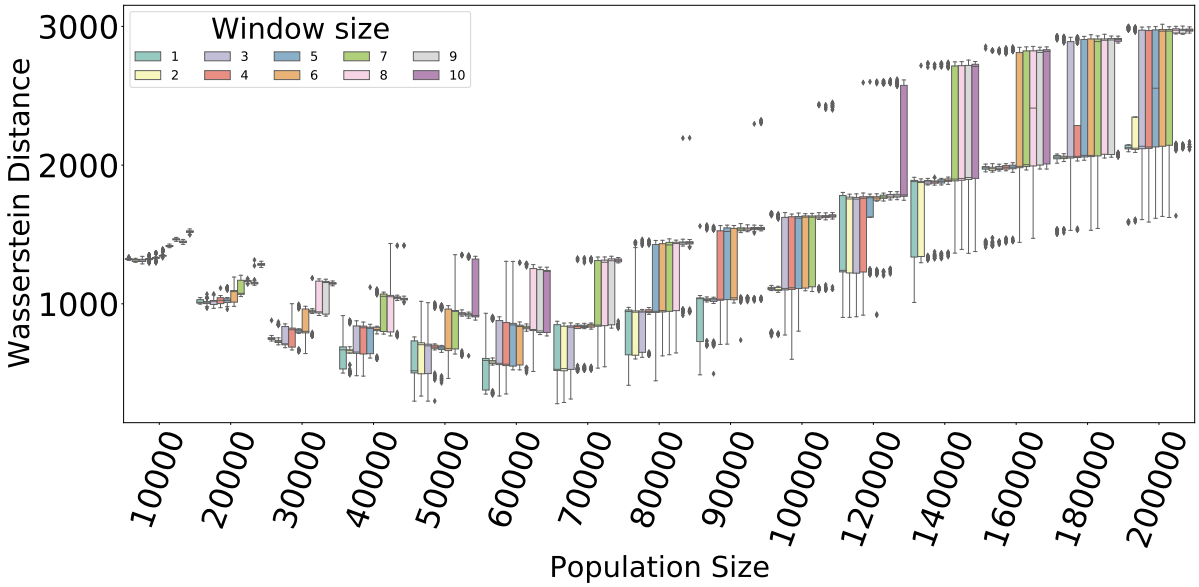

Figure 15. Gaussian distribution: for each population size on the x axis, a group of boxplots displays the Wasserstein Distances (y axis) obtained in the 100 experiments for that population size. Each color represents the window size used for the simulation.

in Figure 16.

8.3 Fitting curve

As explained before, a distinct approach to get an informed estimate of the size B consists of starting from the values observed in Figure 13, in finding a fitting function, and in extrapolating its values.

To do so, after having looked at the data at our disposal, we have observed that, roughly speaking, the amount of new IPs (i.e., never seen so far) observed on a daily basis was decreasing linearly over time. We were thus hoping to be able to find a good fitting function (*Scipy.optimize Curve fit function*, n.d.), thanks to an exponentially decaying one. We found out by means of simulations that the best fit was achieved with the following function:

$$a * (1 - e^{-(x-b)/c}) \tag{1}$$

The parameters that provide the best fit are:

$$a = 2.77313369e + 04$$
$$b = -4.77879543e - 01$$
$$c = 8.04885708e + 01$$

The fitted curve is represented in Figure 17. To assess their similarities, we calculate the Pearson correlation factor (Stigler, 1989) and obtain the value 1.000 which indicates a total positive linear correlation, confirming the adequacy of our fitting function which is visible by the quasi superposition of both curves in Figure 17.

We can now use that fitting function to extrapolate the total amount of distinct IPs we would have seen, had we been able to run the case study for 3 years. Figure 18 shows how the curve reaches a plateau after a bit more than a year. Thus, according to this distinct approach, the bots we have observed only have a couple of tens of thousands of IPs at their disposal, a value that is consistent with the ones found with the first approach.

9. Discussion

The high-level goals of our projects were to have a better comprehensive view of the ecosystems of scraping bots and find new ways to moderate their malicious activities against websites. To achieve these goals, we successfully built a fully functioning honeypot that is capable of serving real-world bot scrapers.

The first case study experienced a complete drop in the traffic after 48h. As explained in Section 5.2, it is quite probable that the bots did submit semantically incorrect requests to detect the honeypot. This showed us how attackers could take

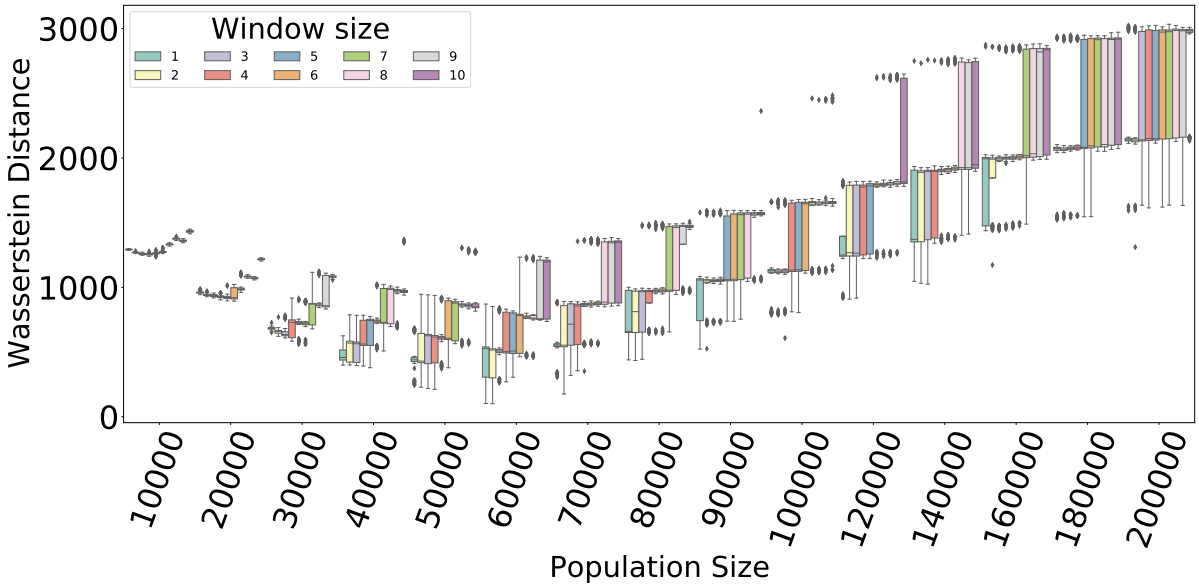

Figure 16. Beta distribution: for each population size on the *x* axis, a group of boxplots displays the Wasserstein Distances (*y* axis) obtained in the 100 experiments for that population size. Each color represents the window size used for the simulation.

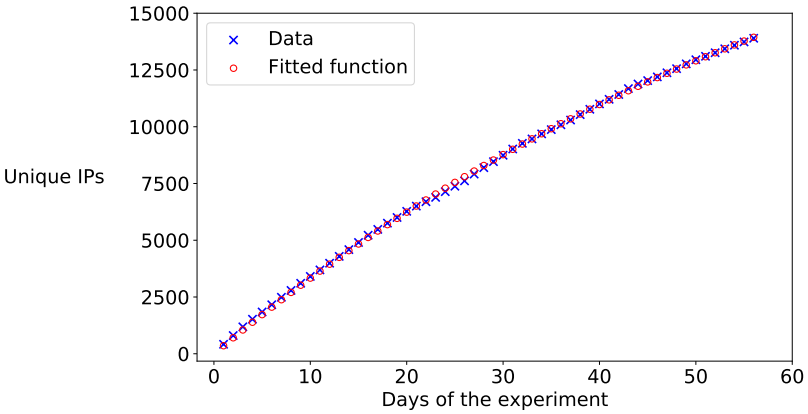

Figure 17. Projection of the real data on the fitting curve values

advantage of corner use cases to investigate the system. Moreover, this outcome helped us in finding and solving this particular inconsistency with the real system.

Furthermore, thanks to it, we realised we should favour studying bot signatures with a regular activity across different days. Moreover, we concluded that redirecting just part of the traffic marked with the same bot signature could bias the case study since the botnet did respond differently when we increased the amount of traffic handled by our honeypot. We applied this earned knowledge to build a second case study.

In the new case study, we did not see any change of behaviour in the traffic before the introduction of COVID-19 restrictions on flights. This gives us confidence that the platform worked properly and it was able to serve the bots with inaccurate prices without being detected.

During the case studies we questioned if it was possible to recognise a bot campaign from the payload information. The very strong correlation among the requests of the second case study, as shown in Section 6, is as close as a ground truth one could hope for. The anti-bot detection solution identifies many IPs as behaving like bots but our experience in looking at the logs gives us no assurance that IPs flagged with a given signature belong to the same botnet. Indeed, the goal of each signature is to fingerprint *"a bot"*, not *"the bot from botnet X, Y or Z"*.

Moreover, semantic similarities observed in the first case study, even if based on a much smaller period of times and requests, indicate that requests associate with a bot signature are likely to belong to the same campaign. These requests were just 10% and then 50% of the total traffic of a particular bot signature. However, since the selection was done randomly, we can affirm with confidence that they are a representative sample of the requests sent with that bot signature.

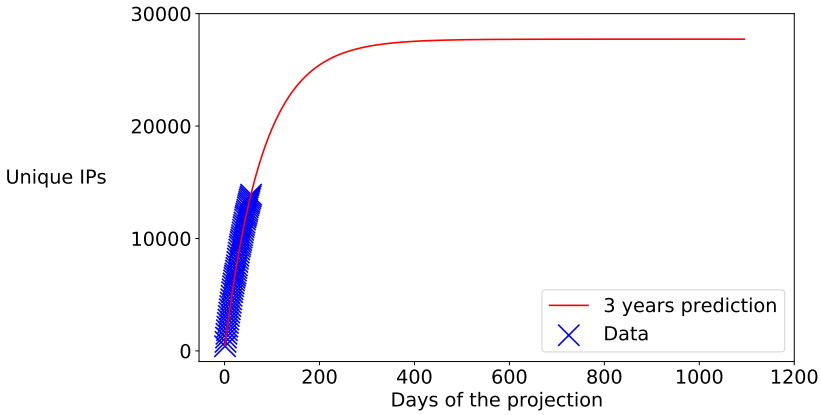

Figure 18. 3 years prediction of the number of different IPs that would have been seen in the honeypot

The visible behavioural correlation found in both case studies gave us the idea to start considering a detection method on top of the fingerprinting. Payload information could be used as a secondary approach to classify together requests coming from the same botnet.

In the second part of our project, we carried out extensive analysis on the IP addresses collected in the second case study. We know these analyses require *clean* data sets to be able to derive meaningful conclusions. We are very well aware though that, compared to all the bots that are out there, our data set is relatively small and we do not pretend that our conclusions can, or should, be extended to all botnets that are in activity. Our results do only apply to the botnet we have studied. Having said so, all elements at our disposal, explained in Section 6.2, Section 7 and Section 8, seem to indicate that this botnet is a perfect example of so-called APBs, Advanced Persistent Bots, and is thus representative of the many others that are scraping websites. Therefore, we have good reasons to believe that our results could be generalised to many other botnets, without having, at the moment the data to support this claim. We hope to see other researchers confirming our findings in the near future.

If true, this would mean that large websites victims of web scraping bots would see the same IPs coming back regularly and that the grand total of IPs they would have to watch for would remain manageable (in the order of tens of thousands instead of tens of millions). An IP blocking strategy could thus be rejuvenated: seeding their sets of IPs with the ones clearly identified as behaving as bots, that strategy could enable us to catch the most evasive bots when they show up with a known bad IP. Redirecting these IPs to a fake website instead of blocking them would also enable us to keep watching their behaviour and, possibly, redirect them to the real website if their requests are not consistent with those of known bots (i.e., in the case of a false positive).

10. Limitations and future works

In this work, we redirect selected traffic, associated with a bot signature, to our honeypot. It is worth noting that, if a real user generates a request which is erroneously detected as a bot one, it is sent to our honeypot. When trying to proceed further in the user journey, by continuing the booking flow on the real site, the user will receive an error message. This is a limitation of our system. However, we chose signatures with optimal levels of specificity and we never observed this type of error message in the normal logs during our case studies.

Our honeypot platform serves modified fares to the bots. Nevertheless, it needs access to the original fares to have a base on which to apply the changes. To achieve this goal, we took advantage of an API of the IT provider. We had reduced access to the API and this limited our choices of bot signatures. In the first case study, since the volume of traffic associated with the bot signature was much more elevated than our threshold, we did not manage to redirect all the requests but just a percentage of it. Moreover, since in the second case study we wanted to divert the whole traffic, we had to find a bot signature whose entire volume was below the API threshold. Even though this added a requirement for the choice of the bot signature, we believe that our work was only partially impacted. Bot campaigns with greater volumes of sent requests are usually already receiving countermeasures and thus they were not already matching the other criteria we decided to apply in the choice.

Another limitation of the honeypot became evident after the behavioural analysis of the first case study. It consists of the lack of coverage of all the possible combinations of parameters in the HTTP payload. A more in-depth study of the correlation of all parameters was needed to have a comprehensive view of all the corner cases. Performing the behavioural analysis of the second case study, we studied the distribution and interaction of all the parameters. We did not find any case of corner case used to detect our platform.

In both our case studies, the requests to be examined were produced from a single bot signature. Moreover, they were

collected only for a finite period of time. Hence, we know that the behavioural analyses and the IP modelling we performed can not pretend to be directly representative of the whole bot traffic. However, they provide a good picture of the behaviour of these particular botnets and a good starting point for future analysis and comparisons.

Finally, some limitations were produced by the initial setup we used. We redirected to the honeypot requests which were produced by bots aware to be targeting a protected system. The IT provider has suffered from bot attacks for many years, using different countermeasures. Thus bots choosing to send requests there know they will encounter some type of countermeasures. This on the other hand tells us that the studied botnets were sophisticated enough to try to overcome leading commercial anti-bot solutions.

The results presented in this paper helped us in convincing our partner, the major IT provider, to move forward into developing a honeypot environment built in their production system and the work is underway. In this solution, cache prices could be served to the bots at a cheap price for the provider. Some sensitivity analysis remains to be done to assess the cache refreshing rate.

Beyond the development of the built-in honeypot, we plan to propose new studies to validate the behavioural analyses of the payloads and the IP blocking solution. Moreover, we are working to assess, at a higher level, the number of IP addresses at the proxies services disposal. Furthermore, we are examining new ways to perform proxies fingerprinting, in such a way to have a better overview on the bots.

We felt it was important to share our preliminary results with the community not only to let other benefit from the gained insights but also, possibly, to obtain feedback on important elements we could have missed. We do hope our contributions will participate in diminishing the negative impact created by these bots on the global Internet ecosystem.

11. Conclusion

In this paper, we analysed the phenomenon of advanced bot scanners by designing and implementing a platform inspired by the traditional honeypots. Thanks to our infrastructure, we have studied specific web scraping botnets that are representative of the plague most airlines' websites are suffering from.

We have shown how stealthy *Advanced Persistent Bots* scrap the web content of an airline company by sending only one request per day and per IP. In the second case study, for 56 days, our honeypot has served to a specific botnet modified prices without being detected. A substantial fraction of these IPs are reused over time and they exhibit a behavioural pattern that gives us high confidence that they all are operated by the same bot master. This opens the way for future work in two distinct directions. First of all, because of the repetition of the requests and the loose verification of the results made by the bots, we can probably serve inexpensive cached results. Secondly, a longitudinal large-scale analysis of suspicious IPs exhibiting the same behaviour may lead to a new, efficient, method to identify IP addresses to be blocked, on top of fingerprinting.

Furthermore, thanks to two distinct mathematical models, we have shown that the total amount of IPs at the disposal of the botnet of the second case study was most likely in the low tens of thousands. We have also given pieces of evidence that these IPs were provided by proxy services, thought to be able to provide tens of millions of IPs to their customers. If our finding applies, as we think it does, to other botnets then an IP-blocking strategy could be applied, contrary to the common belief. We encourage others to carry out similar case studies to confirm or deny, our findings while we are in the process of testing our conjecture in a new large-scale case study.

Declaration of Conflicting Interest

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Acknowledgments

We would like to thank the reviewers for their valuable and insightful comments which helped to improve the quality of the manuscript.

References

Amin Azad, B., Starov, O., Laperdrix, P., & Nikiforakis, N. (2020). Web runner 2049: Evaluating third-party anti-bot services. In *17th conference on detection of intrusions and malware & vulnerability assessment (DIMVA 2020).* Lisboa, Portugal.

The Bait and Switch Honeypot. (n.d.). `http://baitnswitch.sourceforge.net/`. (Accessed: 2020-06-19)

Catakoglu, O., Balduzzi, M., & Balzarotti, D. (2016). Automatic extraction of indicators of compromise for web applications. In *Proceedings of the 25th international conference on world wide web* (pp. 333–343).

Cheswick, B. (1992). An evening with berferd in which a cracker is lured, endured, and studied. In *Proc. Winter USENIX Conference* (pp. 20–24). San Francisco, CA, USA.

Chiapponi, E., Catakoglu, O., Thonnard, O., & Dacier, M. (2020). HoPLA: a Honeypot Platform to Lure Attackers. In *Computer & Electronics Security Applications Rendez-vous, Deceptive security Conference (C&ESAR 2020), part of European Cyber Week.* Rennes, France.

Chiapponi, E., Dacier, M., Todisco, M., Catakoglu, O., & Thonnard, O. (2020). Botnet sizes: when maths meet myths. In *1st International Workshop on Cyber Forensics and Threat Investigations Challenges in Emerging Infrastructures(CFTIC), held in conjuction with the 18th International Conference on Service Oriented Computing (ICSOC 2020).* Dubai, UAE.

Cohen, F. (2006). The use of deception techniques: Honeypots and decoys. *Handbook of Information Security*, *3*(1), 646–655.

Comprehensive IP address data, IP geolocation API and database - IPinfo.io. (n.d.). https://ipinfo.io/. (Accessed: 2020-11-19)

Cosby, Donald J. (2003). *67th district court, tarrant county, texas. cause no. 067-194022-02: American airlines, inc. vs. farechase, inc.* http://www.internetlibrary.com/pdf/american%20airlines%20farechase.pdf. (Accessed: 2021-04-27)

Delong, M., Filiol, E., & David, B. (2019). Investigation and surveillance on the darknet: An architecture to reconcile legal aspects with technology. In *Eccws 2019 18th european conference on cyber warfare and security* (p. 151).

Dietrich, S., Long, N., & Dittrich, D. (2000). Analyzing distributed denial of service tools: The shaft case. In *Proceedings of the 14th usenix conference on system administration* (pp. 329–339). New Orleans, Louisiana, USA.

Djanali, S., Arunanto, F., Pratomo, B. A., Baihaqi, A., Studiawan, H., & Shiddiqi, A. M. (2014). Aggressive web application honeypot for exposing attacker's identity. In *2014 the 1st international conference on information technology, computer, and electrical engineering* (pp. 212–216).

Endicott-Popovsky, B., Narvaez, J., Seifert, C., Frincke, D. A., O'Neil, L. R., & Aval, C. (2009). Use of deception to improve client honeypot detection of drive-by-download attacks. In *International conference on foundations of augmented cognition* (pp. 138–147).

Fraud prevention | detect fraud | fraud protection | prevent fraud with IPQS. (n.d.). https://www.ipqualityscore.com/. (Accessed: 2020-11-19)

Garg, N., & Grosu, D. (2007). Deception in honeynets: A game-theoretic analysis. In *2007 ieee smc information assurance and security workshop* (pp. 107–113).

Haque, A., & Singh, S. (2015). Anti-scraping application development. In *2015 international conference on advances in computing, communications and informatics (icacci)* (p. 869-874). Kochi, India.

Higher Regional Court of Hamburg. (2009). *Ryanair vs Vtours.* (decision dated 28 Mai 2009, file no 3 U 191/08, ECLI:DE:OLGHH:2009:0528.3U191.08.0A)

Imperva. (2019). *How bots affect airlines* (Tech. Rep.). Author. Retrieved from https://www.imperva.com/resources/reports/How-Bots-Affect-Airlines-.pdf

Imperva. (2020). *Imperva bad bot report* (Tech. Rep.). Author. Retrieved from https://www.imperva.com/resources/resource-library/reports/2020-bad-bot-report/

IpInfo.io. (August 2020). Personal communication.

Jung, J., & Sit, E. (2004). An empirical study of spam traffic and the use of dns black lists. In *Proceedings of the 4th acm sigcomm conference on internet measurement* (p. 370–375). Taormina, Sicily, Italy: Association for Computing Machinery.

Labrea: "sticky" honeypot and ids. (n.d.). https://labrea.sourceforge.io/labrea-info.html. (Accessed: 2021-02-21)

Laperdrix, P., Bielova, N., Baudry, B., & Avoine, G. (2020, April). Browser fingerprinting: A survey. *ACM Trans. Web*, *14*(2). Retrieved from https://doi.org/10.1145/3386040 doi: 10.1145/3386040

Leita, C., & Dacier, M. (2008). Sgnet: a worldwide deployable framework to support the analysis of malware threat models. In *2008 seventh european dependable computing conference* (pp. 99–109).

Levina, E., & Bickel, P. (2001). The earth mover's distance is the mallows distance: Some insights from statistics. In *Proceedings eighth ieee international conference on computer vision. iccv 2001* (Vol. 2, pp. 251–256). Vancouver, Canada.

Li, X., Azad, B. A., Rahmati, A., & Nikiforakis, N. (2021). Good bot, bad bot: Characterizing automated browsing activity. In *2021 ieee symposium on security and privacy (sp)* (p. 17).

McKenna, S. (2016). *Detection and classification of web robots with honeypots* (Unpublished master's thesis). Naval Postgraduate School, Monterey, California, USA.

Mi, X., Feng, X., Liao, X., Liu, B., Wang, X., Qian, F., … Liu, Y. (2019). Resident evil: Understanding residential IP proxy

as a dark service. In *2019 IEEE symposium on security and privacy (SP)* (pp. 1185–1201). San Francisco, CA, USA. (ISSN: 2375-1207) doi: 10.1109/SP.2019.00011

Ni, D. (2019). *Top 10 residential, backconnect & rotating proxies for web scraping.* https://www.scraperapi.com/blog/the-10-best-rotating-proxy-services-for-web-scraping/. (Accessed: 2020-08-20)

Nicomette, V., Kaaniche, M., Alata, E., & Herrb, M. (2011). Set-up and deployment of a high-interaction honeypot: experiment and lessons learned. In (Vol. 7, pp. 143–157). Springer.

Nunes, S., & Correia, M. (2010). Web application risk awareness with high interaction honeypots. In *Actas do inforum simposio de informatica (september 2010).*

Pouget, F., & Dacier, M. (2004). Honeypot-based forensics. In *Auscert asia pacific information technology security conference.*

Pouget, F., Dacier, M., & Debar, H. (2003). *White paper: honeypot, honeynet, honeytoken: terminological issues* (Tech. Rep. Nos. EURECOM+1275). Eurecom. Retrieved from http://www.eurecom.fr/publication/1275

Regional Court of Hamburg. (2008). *Ryanair vs Vtours.* (decision dated 28 August 2008, file no 315 O 326/08, ECLI:DE:LGHH:2008:0828.315O326.08.0A)

Samarasinghe, N., & Mannan, M. (2019a). Another look at TLS ecosystems in networked devices vs. web servers. *Computers & Security*, *80*, 1 – 13. doi: https://doi.org/10.1016/j.cose.2018.09.001

Samarasinghe, N., & Mannan, M. (2019b, 07). Towards a global perspective on web tracking. *Computers & Security*, *87*, 101569. doi: 10.1016/j.cose.2019.101569

Scipy.optimize Curve fit function. (n.d.). https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html. (Accessed: 2020-07-21)

Stigler, S. M. (1989). Francis galton's account of the invention of correlation. *Statistical Science*, *4*(2), 73–79. Retrieved from http://www.jstor.org/stable/2245329

Suzuki, K., Tonien, D., Kurosawa, K., & Toyota, K. (2006). Birthday paradox for multi-collisions. In M. S. Rhee & B. Lee (Eds.), *Information security and cryptology – ICISC 2006* (pp. 29–40). Berlin, Heidelberg: Springer Berlin Heidelberg.

Takemori, K., Rikitake, K., Miyake, Y., & Nakao, K. (2003). Intrusion trap system: an efficient platform for gathering intrusion-related information. In *10th international conference on telecommunications, 2003. ICT 2003.* (Vol. 1, pp. 614–619 vol.1). doi: 10.1109/IC.2003.1191480

Thonnard, O., & Dacier, M. (2008). Actionable knowledge discovery for threats intelligence support using a multi-dimensional data mining methodology. In *2008 ieee international conference on data mining workshops* (pp. 154–163).

Thonnard, O., Mees, W., & Dacier, M. (2009). Addressing the attack attribution problem using knowledge discovery and multi-criteria fuzzy decision-making. In *Proceedings of the acm sigkdd workshop on cybersecurity and intelligence informatics* (pp. 11–21).

Tor project. (n.d.). https://www.torproject.org/. (Accessed: 2020-08-14)

Tzu, S. (1971). *The art of war* (Vol. 361). Oxford University Press, USA.

Vastel, A., Rudametkin, W., Rouvoy, R., & Blanc, X. (2020, February). FP-Crawlers: Studying the Resilience of Browser Fingerprinting to Block Crawlers. In O. Starov, A. Kapravelos, & N. Nikiforakis (Eds.), *MADWeb'20 - NDSS Workshop on Measurements, Attacks, and Defenses for the Web.* San Diego, United States. doi: 10.14722/ndss.2020.23xxx

Venema, W. Z. (1992). Tcp wrapper: Network monitoring, access control, and booby traps. In *Usenix summer.*

Von Ahn, L., Blum, M., Hopper, N. J., & Langford, J. (2003). CAPTCHA: Using hard AI problems for security. In E. Biham (Ed.), *Advances in cryptology — EUROCRYPT 2003* (pp. 294–311). Berlin, Heidelberg: Springer Berlin Heidelberg.