HoPLA: a Honeypot Platform to Lure Attackers

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

[1] Eurecom, France
[2] Amadeus IT Group, France
{elisa.chiapponi,marc.dacier}@eurecom.fr
{onur.catakoglu,olivier.thonnard}@amadeus.com

Abstract. Airline websites are the victims of unauthorized 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. They are provided by specialized companies that offer them as "bots as a service" and they leverage professional proxying companies (mis)using millions of residential IP addresses. On top of the valuable information taken away, these huge quantities of requests consume a very substantial amount of resources on the airline websites. In this work, we present a platform capable of mimicking these sites, at a much lower cost, and we provide early results on an experiment in which we have lured for almost 2 months several bots and have fed them indistinguishable inaccurate information.

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 [12]. They have continuously evolved from relatively rudimentary pieces of software to very sophisticated components such as the numerous "all in one sneaker bots" (eg., aiobot.com) that automate the buying process of luxury goods in high demands. To increase their resilience, the bots take advantage of proxying 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 [3,2,4]. These IPs are provided by Residential IP Proxies as a Service, which have been investigated by Mi et Al. in the first comprehensive study of these services [18].

A 2019 Imperva report [6] describes very clearly how the airline industry is particularly impacted by these armies of bots. In 2017, according to that report, the proportion of bad bots traffic to airline 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 behavior, load JavaScript and external assets, tamper with cookies, perform browser automation, give the impression that the mouse is moving on the screen, etc.

Almost all these bots are used to gather free information from the airlines' sites about flights and ticket prices. The conjecture is that unauthorized business

intelligence companies, online travel agencies, and data aggregators are beyond such bots activities because a large part of their business relies on web scraping. They harness information, increasing dramatically the number of requests to be served by airlines' websites while the number of seat bookings remain stable. This increases the so-called "look-to-book" ratio which sales and revenue management departments pay attention to. The damages to the business model, as well as the costs incurred to support these bot requests, explain the explosion of a whole new ecosystem of anti-bots techniques. A very recent publication [7] has shared insights on the design and implementation of 15 popular commercial anti-bot services.

An arms race exists between bot makers and anti-bot providers: as soon as a family of bots is blocked, their operators replace them with a new one which defeats the protection for a varying amount of time. To gain the upper hand against the bots, we propose a platform where identified bots are provided 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.

To present our work, the paper is structured as follows. In Section 2, we describe the problem we try to solve and the contributions we make. Section 3 offers a brief review of the state of the art. Section 4 describes the experimental environment we have used. Section 5 offers and discusses early results. Finally, we conclude in Section 6 with ideas for future work.

2   Problem definition and contributions

We have had the opportunity to look at web sites operated by a major IT provider for the airline industry. These sites are protected by one of the leading commercial anti-bots service, placed in front of them. These services check the origin and the fingerprints associated with each request against a large number of "signatures"[1].

An action (block/captcha[30]/accept) is associated with each signature. It is noteworthy that some requests do match a signature but are allowed to pass through. This is typically the case when the confidence in a bot-issued request is too low. However, as more requests are issued by the same IP, the confidence score can evolve, and, typically after one hundred requests or more, the requester might eventually be blocked, as experimentally observed in [7].

Having observed the competing efforts of bots and anti-bots, 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

---

[1] This is a very simplified explanation due to the lack of space; we refer the interested reader to [28] for more information on the topic.

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 avoid them understand what we do.

In this paper, we address this problem by means of a first experiment that led to the discovery of three important pieces of information:

1. We identify a specific class of super stealthy bots that are characterized by an extreme distribution of their activity. Most of the time, they only send a single request per day and per IP, and almost never more than two.
2. We show that a behavioral pattern emerges when looking at all these requests put together. 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.
3. We show that it is possible to provide inaccurate information without being detected by the bot operators.

3   State of the art

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"[27]. 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, 1992, B. Cheswick told us about his "evening with Berferd" [9] and, just after that, W. Venema made it possible for everyone to play with the idea with the creation of TCPWrapper [29]. In [10], Cohen formalises the notion of deception in computers and reviews the early works on honeypots. [15] provides a game-theoretical view of these deception approaches.

Since then, *honeypots*, *honeynets* and *honeytokens* [21] have received a lot of attention. They exist in all kinds of flavors, low/mid/high interaction [16]. 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 [22,19] or derive actionable knowledge about them [24], possibly leading to attack attribution applications [25]. Others take an active role in slowing down the attackers. They are then usually called sticky honeypots, tarpits or crawler traps [5,1,11]. Web application honeypots, in particular, have received a lot of attention, with, among many others, the following interesting pieces of work [20,8,14,13]. In particular, when it comes to bot detection, [17] provides a good survey on the various works that have tried to use web-based honeypots and honeytokens against them.

Our work has been inspired by the *Bait&Switch* Honeypots [5] and the Intrusion Trap Systems [23] where the authors redirect malicious traffic to a honeypot that mirrors the real site under protection. The main differences 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.

4   Experimental setup

We have carried out a 56 days experiment in collaboration with a major IT provider for airlines websites. The provider offers different products for the booking process. We focused our attention on a specific one, product A, for which the customer airlines build their own private website. On this domain, a user can insert information about the flight to search, such as locations and dates for the departure and return flight (if any). When the user finalizes the process, a HTTP POST request is built. Subsequently, the request is sent to the booking domain, which belongs to the IT provider.

As shown in Fig. 1, the requests do not go directly to the booking website but they pass through an anti-bot commercial solution. This artifact is used to recognize 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[30], a JavaScript challenge, or blocking the bot.

In our setup, we implemented a new type of custom action, which consists in forwarding the request to a honeypot, as shown with a dash-line in Fig. 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 is able to modify the fares, insert them into the structure and send the response back. In this way,
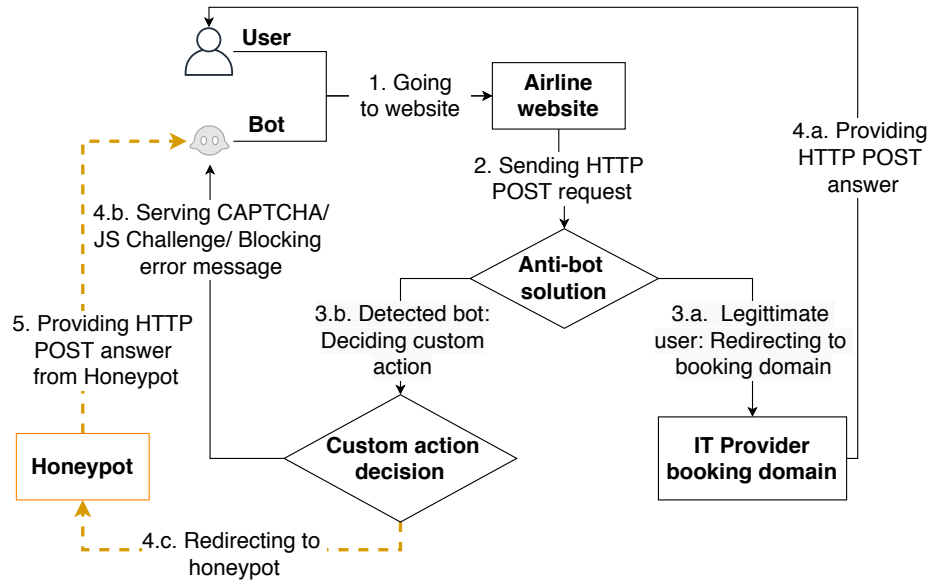
Fig. 1. Scheme of the booking system. The dashed line shows the new addition of the honeypot into the flow.

bots receive a response with the same syntax but different prices with respect to the original one.

It is worth noting that, if a real user generates a request which is 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. We have never observed such an error message in the normal logs during our experiment.

This work was carried out in collaboration with a specific airline company, which uses product A. In general, 1 million requests are sent daily to that airline website. The anti-bot solution usually blocks close to 40% of them.

Among the non-blocked requests, we noticed that some were labeled by a signature from the anti-bot. That signature was triggered every day, only once, during a 40 minutes window. In the period preceding the experiment, these matching requests were almost never seen completing a booking. All these elements led us to conjecture that all requests having this signature were likely false negative of the anti-bot and we have decided to investigate them further.

The anti-bot was configured in such a way that all the requests matching that signature would be forwarded to our honeypot. During the first three days of the experiments, we did provide the correct answers, to ensure that our modified setup did not generate any noticeable artefact that would have scared the bots away.

After this initial observation period, we systematically increased the ticket prices by 5%, for a random selection of 10% of the requests. Due to this strategy, an observer could have noticed an anomalous fluctuation of the price in specific points in time. For example, if multiple requests with the same parameters were made in a short amount of time and just one was randomly chosen for the increase, this should have been recognised as an anomaly by a careful observer. Our goal was to understand if the bot master was actually checking for such a pattern and if a small modification of the price would have been caught.

5   Results

5.1   Introduction

In this section, we will present the results of our experiment. First, in subsection 5.2, we will provide an overview of the retrieved traffic. Subsection 5.3 will show our study of the information about flights requested by the bots.

5.2   General analysis

Between the 07/01/2020 and the 02/03/2020, the honeypot received 22,991 requests, each day within the same 40 minutes window. There was no change in behavior before and during our experiment. Changing randomly some of the values without causing their departure, we have learned that i) they do not have a ground truth to compare the returned values and ii) their plausibility check is not sophisticated enough to detect small changes, not even by correlating values collected over several days.

The matching requests have all disappeared since March 3rd, which corresponds to the day after the first COVID-19 quarantine decisions were made by the government of the country where this airline company is based. For this reason and the fact that the bot behavior did not change after the increase of the price, we believe the honeypot was not detected by the bots during the experiment.

The requests came from 13,897 different IP addresses. 88% of them made a single request per day and 97% less than 2 (see Fig. 2)! 32% of these IPs appeared on more than 1 day which is, somehow surprising, knowing that some proxy services offer several millions of distinct IP addresses.

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.

TOR exit nodes To better understand if the IPs collected in the honeypot were involved in malicious activities during the time of the experiment, we checked if they were used in the Tor network [26]. We analyzed 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 in 5 different days. In Table 1 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. Our hypothesis is that the bots tried to take advantage of the tor nodes but they did not continue with this strategy.
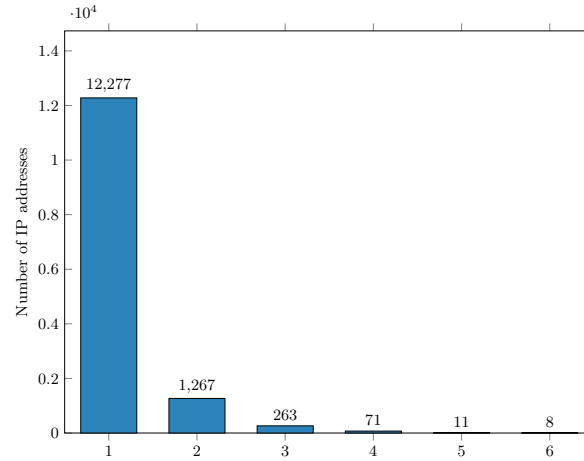

Fig. 2. # of IP addresses having sent at most X requests per day

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

5.3   Behavioral analysis

We decided to study the payloads of the bot requests, something the anti-bot is not looking at. 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.

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 Fig. 3, 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 experiment. 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 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.

Fig. 4 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 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 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 experiment, to be compared with the average 410 requests per day.

Tuples statistics 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 experiment. 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 span 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 behavior 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 bot did not found 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.
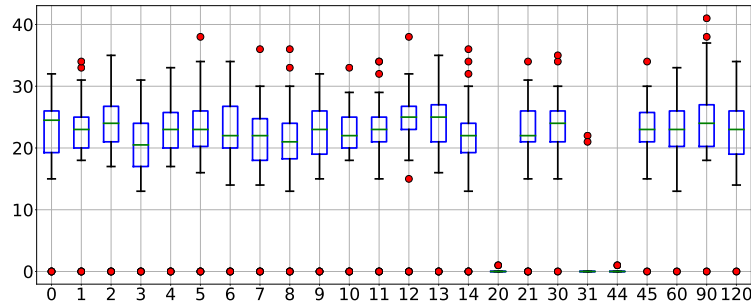

Fig. 3. Boxplots representing the distribution of the intervals among different paths
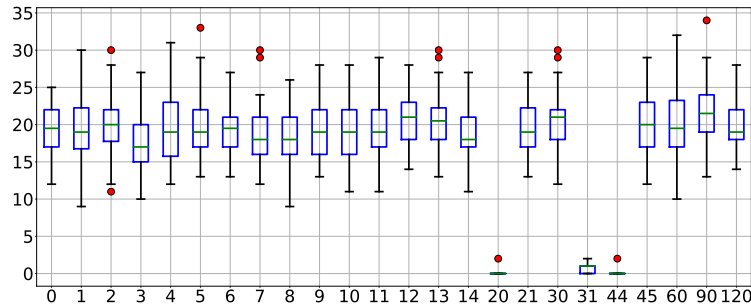

Fig. 4. Boxplots representing the distribution of the intervals among different request dates

6   Discussion and Conclusion

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. For 56

days, our honeypot has served them modified prices without being detected. A substantial fraction of these IPs (32%) are reused over time and they exhibit a behavioral 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. Some sensitivity analysis remains to be done to assess the cache refreshing rate. 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.

References

1. Labrea: "sticky" honeypot and ids, `https://labrea.sourceforge.io/labrea-info.html`
2. Luminati, `https://luminati.io/`
3. Oxylabs, `https://oxylabs.io/`
4. Smart Proxy, `https://smartproxy.com/`
5. The Bait and Switch Honeypot, `http://baitnswitch.sourceforge.net/`
6. How bots affect airlines (2019), `https://www.imperva.com/resources/reports/How-Bots-Affect-Airlines-.pdf`
7. Amin Azad, B., Starov, O., Laperdrix, P., Nikiforakis, N.: Web runner 2049: Evaluating third-party anti-bot services. In: 17th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2020). `https://hal.archives-ouvertes.fr/hal-02612454`
8. Catakoglu, O., Balduzzi, M., Balzarotti, D.: Automatic extraction of indicators of compromise for web applications. In: Proceedings of the 25th International Conference on World Wide Web. pp. 333–343 (2016)
9. Cheswick, B.: An evening with berferd in which a cracker is lured, endured, and studied. In: Proc. Winter USENIX Conference, San Francisco. pp. 20–24 (1992)
10. Cohen, F.: The use of deception techniques: Honeypots and decoys. Handbook of Information Security **3**(1), 646–655 (2006)
11. Delong, M., Filiol, E., David, B.: 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. Academic Conferences and publishing limited (2019)
12. Dietrich, S., Long, N., Dittrich, D.: Analyzing distributed denial of service tools: The shaft case. In: LISA. pp. 329–339 (2000)
13. Djanali, S., Arunanto, F., Pratomo, B.A., Baihaqi, A., Studiawan, H., Shiddiqi, A.M.: 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. IEEE (2014)
14. Endicott-Popovsky, B., Narvaez, J., Seifert, C., Frincke, D.A., O'Neil, L.R., Aval, C.: Use of deception to improve client honeypot detection of drive-by-download attacks. In: International Conference on Foundations of Augmented Cognition. pp. 138–147. Springer (2009)
15. Garg, N., Grosu, D.: Deception in honeynets: A game-theoretic analysis. In: 2007 IEEE SMC Information Assurance and Security Workshop. pp. 107–113. IEEE (2007)

16. Leita, C., Dacier, M.: Sgnet: a worldwide deployable framework to support the analysis of malware threat models. In: 2008 Seventh European Dependable Computing Conference. pp. 99–109. IEEE (2008)
17. McKenna, S.: Detection and classification of web robots with honeypots. Naval Postgraduate School, Monterey, California (2016)
18. Mi, X., Feng, X., Liao, X., Liu, B., Wang, X., Qian, F., Li, Z., Alrwais, S., Sun, L., Liu, Y.: Resident evil: Understanding residential IP proxy as a dark service. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 1185–1201 (2019). https://doi.org/10.1109/SP.2019.00011, ISSN: 2375-1207
19. Nicomette, V., Kaaniche, M., Alata, E., Herrb, M.: Set-up and deployment of a high-interaction honeypot: experiment and lessons learned. vol. 7, pp. 143–157. Springer (2011)
20. Nunes, S., Correia, M.: Web application risk awareness with high interaction honeypots. In: Actas do INForum Simposio de Informatica (September 2010). Citeseer (2010)
21. Pouget, F., Dacier, M., Debar, H.: White paper: honeypot, honeynet, honeytoken: terminological issues, http://www.eurecom.fr/publication/1275
22. Pouget, F., Dacier, M., et al.: Honeypot-based forensics. In: AusCERT Asia Pacific Information Technology Security Conference (2004)
23. Takemori, K., Rikitake, K., Miyake, Y., Nakao, K.: 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. https://doi.org/10.1109/IC.2003.1191480
24. Thonnard, O., Dacier, M.: 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. IEEE (2008)
25. Thonnard, O., Mees, W., Dacier, M.: 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 (2009)
26. Tor project, https://www.torproject.org/
27. Tzu, S., Tzu, S., Sun, W., Vu, S.C., et al.: The art of war, vol. 361. Oxford University Press, USA (1971)
28. Vastel, A., Rudametkin, W., Rouvoy, R., Blanc, X.: Fp-crawlers: Studying the resilience of browser fingerprinting to block crawlers. In: NDSS Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb'20) (2020)
29. Venema, W.Z.: Tcp wrapper: Network monitoring, access control, and booby traps. In: USENIX Summer (1992)
30. Von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: Captcha: Using hard ai problems for security. In: International conference on the theory and applications of cryptographic techniques. pp. 294–311. Springer (2003)