BLACKLIST FILTERING FOR SECURITY RESEARCH: BRIDGING THE GAP BETWEEN DOMAIN BLACKLISTS AND MALICIOUS WEB CONTENT

BY

KAISHEN WANG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Adviser:

Associate Professor Michael Donald Bailey

ABSTRACT

Blacklists are a collection of origins (URLs, domains or IP addresses) associated with malevolent activities like the dissemination of malware, facilitation of command and control (C&C) communications, and delivery of spam messages. Blacklists are a simple and convenient way to protect users from these known malicious websites. Although blacklists are not designed for security research, they are commonly used in security research projects as a scan target list for retrieving malicious web contents. However, domain blacklist scans are noisy, as a large portion of websites scanned do not perform malicious activities during the time they are visited. Many blacklisted websites are offline or parked, even though some of them may have contained malicious contents before. Consequently, blacklists cannot be used out-of-box for security research. Kuhrer et al. [1] evaluated the effectiveness of major blacklists in 2014, and proposed a heuristic mechanism to collect training data to build a machine learning classifier to identify parked domains in blacklists. They found up to 10.9% of entries are parked. In this work, we reproduced their approach and found that most of the heuristics and features they used have become stale after five years.

We modernized the prior work approach to identify offline domains and parked domains in blacklists. First, we built and open-sourced an efficient blacklist filter, MGRAB, that can filter domains at several layers: domains with no DNS resolution, closed TCP ports, or error HTTP response code. Using MGRAB, we found that only 43% of all domains in our blacklist (aggregated from 27 domain blacklists) have valid IPv4 address and only 40% of the total domains can be successfully grabbed. Second, we implemented an updated mechanism to detect parked domains using new heuristic strings and created a new random forest classifier. Using the updated mechanism, we found that around 4% of the successfully grabbed domains are parked. Overall, only 33% of the total domains contain meaningful content. Researchers can use MGRAB and the parked domain detection methodology to filter blacklisted website scans for future security research.

ACKNOWLEDGMENTS

I dedicate my work to Zane Ma, who continuously guided me and helped me during every stage of the project. His contribution to this project is significant, indispensable and irreplaceable.

I want to thank my advisor Professor Bailey, who offered me invaluable guidance on the project and spiritual support as well.

I would also thank my beloved parents and other family members for supporting my study overseas, both mentally and financially. My every achievement is impossible without them.

I want to thank my ex-girlfriend, Doris. She accompanied me for a large part of my graduate life and shared much happiness with me, although I didn't write a single line of code related to this project before she returned back to China.

Finally, I want to thank all the members in NSRG, especially Yi and Simon, who offered me help in setting up environment for experiments.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 BACKGROUND 3
2.1 DNS and ZDNS 3
2.2 ZMAP 3
2.3 Banner Grabber, HTTP and ZGRAB 4
2.4 Parked Domain 4

CHAPTER 3 RELATED WORK 5

CHAPTER 4 MGRAB 6
4.1 Motivation and Overview 6
4.2 Design and Implementation 7
4.3 Performance 11
4.4 Validation on DNS Resolution 11

CHAPTER 5 MEASUREMENT RESULT 18
5.1 Measurement Summary 18
5.2 Measurement Result from MGRAB 18
5.3 Parked Domain Detection 19

CHAPTER 6 LIMITATIONS 25

CHAPTER 7 FUTURE WORK 26

CHAPTER 8 CONCLUSION 27

REFERENCES 28

APPENDIX A RESULT OF OLD IDENTIFIERS 31

APPENDIX B RESULT OF NEW IDENTIFIERS 33

CHAPTER 1: INTRODUCTION

Blacklists are a collection of origins (URLs, domains or IP addresses) associated with malevolent activities like the dissemination of malware, facilitation of command and control (C&C) communications, and delivery of spam messages. Security researchers continuously work on detecting these malicious activities and collect the origins (URLs, domain or IP address) of such malevolent activities into "blacklists". Blacklists are a simple and convenient way to protect users from these known malicious websites. For example, Google Safe Browsing [2] is a blacklist that protects over three billion devices every day by showing warnings to users when they attempt to navigate to dangerous sites or download dangerous files. Although there are more advanced dynamic methodologies in detecting malicious websites, Internet users can easily benefit from blacklist as it is easily transferable, simple to use, and does not rely on any software or framework.

Although blacklists are not designed for security research, they are commonly used in security research projects as a scan target list for retrieving malicious web contents. In this study, we examine the gap between scanning domain blacklists and the malicious web contents that may be used by security researchers. Several factors account for this contrast. First, prior work in 2014 demonstrated that blacklists are a noisy scan target data source for security research as a large portion of websites in it do not perform malicious activities during the time they are visited[1]. Many websites in the blacklists are offline or parked although some of them may have contained malicious contents before. Second, researchers usually use banner grabbers like ZGRAB to collect web contents; however, these grabbers are easily detectable and finger-printable, and this allows malicious websites to perform "web cloaking" [3] which presents different content to the banner grabber scan compared to visiting them through browsers. Consequently, result of running a banner grabber against the blacklists cannot be used out-of-box for security research.

Kuhrer et al.[1] evaluated the effectiveness of major blacklists in 2014. They proposed a mechanism of heuristic strings and machine learning techniques to identify parked domains in blacklists and found up to 10.9% of entries are parked. We reproduced their work and found that most of their heuristic strings and some of the features for their machine learning model used to identify parked domains are stale. Only 0.04% of the latest blacklist entries were found to be parked using their old heuristic strings. We suspected that this reduction reflects the inaccuracy of the existing methodology, rather than reflecting an actual absence of parked domains.

We modernized the approach to identify offline domains and parked domains in blacklists.

First, we built and open-sourced a blacklist filter, MGRAB[1], that could filter domains that do not have valid IPv4 address and domains without certain network port open. MGRAB is a general banner grabber that can save the information gathered at every step: DNS traces and resolution of domains, TCP port responsiveness, and the HTML pages of domains. MGRAB outperforms the native ZGRAB (often used for the purpose of a HTTP banner grabber) by 34% for the blacklist we used. Second, we proposed an updated mechanism to detect parked domains using new heuristic strings and a new machine learning model. We extracted 30 new heuristic strings and switched to a new machine learning model with updated features selected.

Using the blacklist filter MGRAB, we found that only 43% of the 10,292,131 domains have valid IPv4 addresses and only 40% of the 10 million domains can be successfully grabbed. Using the updated mechanism, we found around 4% of successfully grabbed domains are parked. We also observed that 14% of the successfully grabbed domains are just error pages. Overall, only 33% of the total domains contain meaningful content. Researchers can use MGRAB and the parked domain detection methodology to generate a pre-filter of blacklist for security research.

[1] `github.com/kwang40/pipelineWrapper`

CHAPTER 2: BACKGROUND

2.1 DNS AND ZDNS

The Domain Name System (DNS) is a central part of the Internet, providing a way to match names to IP addresses. Everything connected to the Internet - laptops, tablets, mobile phones, websites - has an Internet Protocol (IP) address made up of numbers. A website might have an IPv4 address like 172.217.4.46, but this is obviously not easy to remember. However a domain name such as google.com is something people can recognize and remember. DNS maps domain names with IP addresses enabling humans to use memorable domain names while computers on the Internet can use IP addresses.[4]

When a host sends out a DNS query, the query will first go to a local DNS server, which is normally a recursive resolver. If the recursive server does not have the answer in its cache, it will then take over the query and send it to the root name servers. There are 13 root name servers, and they will direct the query to the name servers on next level based on the TLD (top-level domain) of the query. The query will be transmitted between different name servers until it reaches a name server that knows the IP address to the domain in query. Finally, the recursive server will give the original host the answer and store the information in its local DNS cache.

ZDNS[5] is a utility for performing fast DNS lookups, such as completing an A lookup for all names in a zone file, or collecting CAA records for a large number of websites. ZDNS can perform recursive name resolution and supports A, AAAA, ANY, AXFR, CAA, CNAME, DMARC, MX, NS, PTR, TXT, SOA, and SPF records.[6] Instead of operating against a recursive resolver (e.g., an organizational DNS server), ZDNS could also perform its own recursion internally. In this case, ZDNS will round-robin between the 13 published root servers (e.g., 198.41.0.4) and use local cache to improve performance. In our experiment, we observed that ZDNS could perform DNS lookups on 100K domains random sampled from the blacklist in 85 seconds.

2.2 ZMAP

ZMAP is a fast single packet network scanner designed for Internet-wide network surveys. On a computer with a gigabit connection, ZMAP can scan the entire public IPv4 address space in under 45 minutes. With a 10 gigE connection and PF_RING, ZMAP can scan the IPv4 address space in 5 minutes.[6] One key point of its performance it that ZAMPs

receiving code is stateless with respect to the sending code, a valid SYN-ACK that comes back any time before the scan completes will be recorded as a listening host. In order to efficiently support address exclusion through the use of radix trees, ZMAP utilizes a trie specifically designed to handle ranges and frequently used by routing tables.[7]

2.3 BANNER GRABBER, HTTP AND ZGRAB

Banner grabbing is a technique used to gain information about a computer system on a network and the services running on its open ports. Administrators can use this to take inventory of the systems and services on their network. However, an intruder can use banner grabbing in order to find network hosts that are running versions of applications and operating systems with known exploits. [8]

Some grabbers like ZGRAB could perform banner grabbing on HTTP service, retrieving website information together with HTML pages. This information could be useful in many measurements and security related research. ZGRAB[9] is a stateful application-layer scanner that works with ZMAP. ZGRAB is written in Go and supports HTTP, HTTPS, SSH, Telnet, FTP, SMTP, POP3, IMAP, Modbus, BACNET, Siemens S7, and Tridium Fox. For example, ZRGAB can perform a TLS connection and collect the root HTTP page of all hosts ZMAP finds on TCP/443.[6]

2.4 PARKED DOMAIN

Domain parking refers to the registration of an internet domain name without that domain being associated with any services such as e-mail or a website. This may have been done with a view to reserving the domain name for future development, and to protect against the possibility of cybersquatting. Since the domain name registrar will have set name servers for the domain, the registrar or reseller potentially has use of the domain rather than the final registrant.[10]

CHAPTER 3: RELATED WORK

Blacklists play an important role in security related research. When studying behaviors of malicious websites, researchers often leverage existing blacklists as a source of malicious activities. Based on those blacklists, they may detect features in malicious websites and build predictive models that could discover new malicious websites. Features could be extracted from analysis of network traffic [11], [12], inspection of web content [13], [14], URL scrutiny [15], or a combination of those techniques [16]. As a result, improving the quality of blacklist is beneficial for the area of security research.

The banner grabber we built, MGRAB, can provide DNS information of every domain. Such information is not available through most banner grabbers like ZGRAB. DNS information is useful in identifying malicious websites. A survey [17] categorizes many existing works (like [18], [19] and [20]) to discuss the important challenges that the research community should address in order to fully realize the power of DNS data analysis to fight against attacks leveraging malicious domains.

Some researchers conducted research on the quality of blacklists. For example, Gao et al. [21] and Thomas et al. [22] looked at blacklists in Twitter. Similarly, Sinha et al. [23], Rossow et al. [24], and Dietrich et al. [25] evaluated the strength of blacklists in the context of email spam, while Sheng et al. [11] analyzed phishing blacklists. In 2014, Kuhrer et al.[1] also discussed the effectiveness of malware blacklists. They proposed a mechanism of heuristic strings and machine learning techniques to identify parked domains in blacklists and found up to 10.9% of entries are parked. We reproduced their work and found that most of their heuristic strings and some of the features for their machine learning model used to identify parked domains are stale. Only 0.04% of the latest blacklist entries are found to be parked using their old heuristic strings. We believe this number is inaccurate and the methodology needs update. We grounded much of our work of parked domain detection on this paper and updated their methodology. The details can be found in section 5.3.

CHAPTER 4: MGRAB

4.1 MOTIVATION AND OVERVIEW

In many research projects related to analyzing malicious websites behaviors, banner grabbing is very useful as it will provide researchers with data about a large count of malicious websites. ZGRAB is widely used for the purpose of a HTTP banner grabber. For the study, we collected around 10 million malicious URLs from several blacklists. As for the 10 million malicious websites, it will take ZGRAB more than 24 hours to complete grabbing all their information at once. As the main bottleneck of ZGRAB is solving DNS resolution for those domains, this time cost can be reduced to less than 5 hours using a private DNS server. We observed that ZGRAB can be further improved in two aspects. We designed and implemented MGRAB, a pipeline constituted by our modified ZDNS, modified ZMAP and modified ZGRAB.

The first improvement is performance. The lifetime of many malicious websites is relatively short. Many domains from blacklists cannot be resolved at DNS level or their TCP port are closed soon after they appeared on a blacklist. Due to the high ratio of these unavailable websites, ZGRAB's efficiency is not optimized for blacklists. However, there is a more efficient way to discover these unavailable websites. In MGRAB, we use ZDNS to filter out websites without valid DNS resolution and ZMAP to filter out websites without certain network port open. MGRAB also utilizes the DNS resolution obtained from ZDNS to perform the banner grabbing job. MGRAB can perform banner grabbing on URLs from malware blacklist 1.34x faster than the native ZGRAB, and can complete the whole scan of 10,523,269 URLs in 3.5 hours.

The second improvement is that MGRAB collects more information from the pipeline. In addition to the HTML pages grabbed by ZGRAB, MGRAB also collects full recursive DNS traces for every domain through ZDNS (such as the name server every domain belongs to) and the domains/IPs with certain port open through ZMAP. This information is also beneficial for some security research projects. For example, using this information about domains in a blacklist, researchers can understand at which layer (DNS layer, host layer or application layer), that a domain become unavailable. As we have mentioned in section 3, a lot of research projects use DNS information as important features to identify malicious domains.

We validated the correctness of MGRAB. One major concern we had for MGRAB is the DNS resolution of websites change between when they are scanned by different components

in the pipeline. First, we verified that the success counts of grabbing using MGRAB and ZGRAB are almost the same. We then ran a few experiments and analyzed the stability of DNS resolution of malicious websites from blacklist. We found that the DNS resolution of those websites are overall quite stable and the number of possible failures caused by DNS resolution change during the pipeline is tolerable compared to the total amount of data we process. More details are available in section 4.4.

4.2 DESIGN AND IMPLEMENTATION

MGRAB is a tool of general purpose for efficiently collecting information about DNS, port and banner. MGRAB works especially well with list that contains many unavailable domains, like malware blacklists. MGRAB is a pipeline based on ZDNS, ZMAP and ZGRAB. Users can use MGRAB and replace ZGRAB with any banner grabber.

We designed MGRAB to support streaming because of the following two reasons: 1.We want to reduce the time difference between when we get the DNS resolution for an URL and the time we feed that IP into ZGRAB, the final banner grabber. 2. Enabling multiple parts from pipeline to run simultaneously makes the pipeline more efficient.

The major challenge we met is that one domain could be related to multiple urls and one IPv4 address could be related to multiple domains (this common scenario will be elaborated in a later section). It is challenging to achieve the goal that the pipeline will cope with every URL exactly once, as we need to record and retrieve the mapping relationship between URLs, domains and IPv4s from different parts in the pipeline. The connection of different components in MGRAB is illustrated in Figure 4.1 and Figure 4.2.

MGRAB is implemented in golang. It invokes the modified ZDNS and ZMAP as subprocesses and takes care of the input and output of these subprocesses. It outputs the tuple of URL and IPv4, which could be fed into modified ZGRAB. The MGRAB code and all the modified versions of ZGRAB, ZMAP and ZGRAB are open-sourced and available on Github. While implementing and testing MGRAB, we also made contribution to the open-source software tools we used. We fixed a bug in ZDNS and added a feature in ZCRYPTO. All the pull requests have been merged.

4.2.1 Domain To URL

The streaming process will not start until map ① Domain2URL is fully populated. There are two reasons here. First, for the 10 million URLs, it takes only around 20 seconds to read the data and build the map, but it could save more than 2% of DNS lookups where

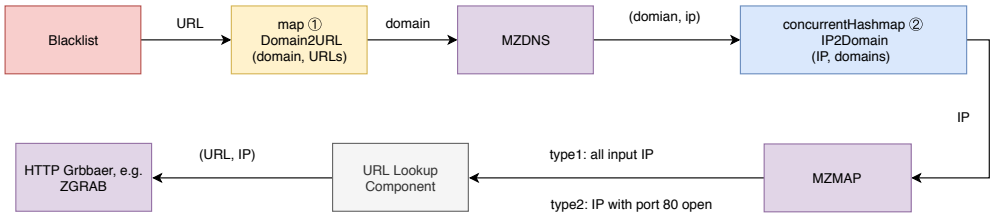

Figure 4.1: **Full Pipeline Flow** - MGRAB relies on modified ZDNS and modified ZMAP and uses several internal map to record the relationship between domains, URLs and IPv4 addresses, e.g. map ① Domain2URL records all the URLs (value) belong to the same FQDN (key), map ② IP2Domain records all the domain (value) belong to the same IP (key).

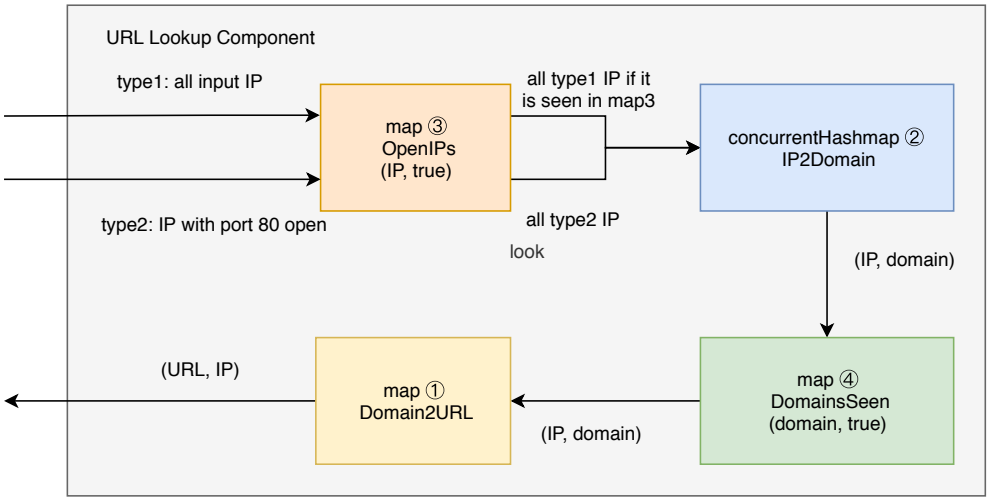

Figure 4.2: **URL Lookup Component Details** - URL Lookup Component takes all output from modified ZMAP and emits every valid URL (with IPv4 address and certain port open) along with its IPv4 address. Four different maps ①, ②, ③ and ④ are used. Map ③ OpenIPs records all the IP with certain port open, and map ④ DomainsSeen records all the domain (value) appeared before.

multiple URLs share the same domain, which is more than 4 minutes. Second, because we want to read from map ① Domain2URL from a later component, streaming requires support concurrent reading and writing for map ① Domain2URL and introduces additional time cost.

4.2.2 Modified ZDNS

As for ZDNS, its output is available in stream. However, it will cost extra work to marshal the result in JSON format, save it to file and unmarshal it again. After the modification, MZDNS will have two goroutines for output. One goroutine will save the full resource records to a file in JSON format, another goroutine will emit (domain, IP) tuples to standard output, which is connected to the next component in the pipeline.

4.2.3 IP2Domain

We need a data structure to memorize the relationship between IPs and domains. As we have explained, multiple domains could share the same IP address. As a result, it is possible to have two components(two threads) read and write on the same key simultaneously. As a result, we need to use a concurrent hashmap [26] ② IP2Domain to store the information.

4.2.4 Modified ZMAP

There are two modifications we made to the native ZMAP.

First, native ZMAP does not support read from stream. In order to improve the performance, native ZMAP will build a trie on all the IP address it wants to probe before actually sending out any probing packets. We disabled such mechanism and made it support reading stream from standard input.

Second, ZMAP only outputs every IP address once even if it sees the same IP address from input for multiple times. Because network packets could be dropped, ZMAP may send more than one packets to probe an IP address. Memorizing the number of times it sees on every input IP address will be expensive and violates its stateless attribute. Modified ZMAP will have two types of output. Type one output will be exactly the same as input, except we append a '#' character as prefix in order to make it distinguishable from type two. Type two output will be the original output, which indicates the specified port (TCP port in our case) is open. For example, if ZMAP sees IP ("172.217.4.46"), it will output "#172.217.4.46" as type one output. If it ZMAP finds that this IP address has port 80 open, it will also output

"172.217.4.46" as type two output. All the type one and type two output will be consumed by the URL Lookup Component.

4.2.5 URL Lookup Component

Inside the URL Lookup Component, a hashmap will record all the type two IPs it has seen. For every type one IP coming, if such IP address has been recorded in the hashmap ③ OpenIPs, it means that we have already probed this IP address and we know that it has the specified port open. All the type one IPs in this case together with all the type two IPs, will go to the next step.

In the next step, for every IP, we can get all the domains related to it through concurrent hashmap ② IP2Domain, and use another hashmap ④ DomainsSeen as a filter to emit every domain only once. Finally, we will find all the URLs related to a domain in map ① Domain2URL and output tuples of (IP, URL).

In general, MGRAB can read a list of URLs, and output all the URLs with their IPs if the FQDN (Fully Qualified Domain Name) of that URL has a valid IPv4 address and that IPv4 address has specified port open. It could save the complete resource records of all the domains with the IPs/domains that have specified port open.

4.2.6 Modified ZGRAB

The native ZGRAB only supports grabbing the root page of domains. We modified it to support grabbing any page of a domain given URL. In MGRAB, ZGRAB can be replaced by any banner grabber, such as Headless Chromium[27] or Telnet[28].

4.2.7 Alternative Approach

In our previous design, instead of using a program to wrap different components together, we used piping and redirection in Linux to connect different parts. Because we need to set and read from map ① Domain2URL and map ② IP2Domain in different components, we put them in Redis stores[29]. However, in experiments we found that Redis is not fast enough and becomes the bottleneck in the pipeline.

We changed to using a golang program to wrap up different components and have full control over their input and output. Inside the golang wrapper program, we use local hashmap and concurrent hashmap for map ① Domain2URL and map ② IP2Domain, which are much faster than Redis.

4.3 PERFORMANCE

We identified the major constraint of ZGRAB (both native ZGRAB and our modified ZGRAB) in our testing environment - the upper bound it can communicate with local DNS server. Because we allow up to 10 redirects for ZGRAB, it still needs to look up DNS resolution through DNS servers even we provide the IP addresses. We use bind9 on another machine to serve as a private DNS server for our measurement and test.

First, we used the top 100K domains from [30] as the source and found that MGRAB has similar performance compared to native ZGRAB in this case. It takes native ZGRAB 435 seconds, and MGRAB 442 seconds to complete all the work. Because the rate of a top domain has successful DNS resolution and TCP port open is very high, the benefits of MGRAB are not as dominant. During the test, we verified that ZGRAB is still the bottleneck in MGRAB in this case. The extra 7 seconds comes from delay of launching the pipeline, ZDNS and ZMAP. Under any circumstances, MGRAB will not be more than 10 seconds slower than the native ZGRAB. However, if we don't use the private DNS server, MGRAB will still be beneficial as ZDNS in MGRAB could provide DNS resolution much faster than from the default DNS server we used (operated by University of Illinois).

As for sparse lists like blacklist, as we expected, performance of MGRAB is improved compared to native ZGRAB. We randomly sampled 100k URLs from the malicious website blacklists. Without using private DNS server, native ZGRAB needs more than 12 minutes to grab all the valid http/https pages, and MGRAB needs 4 minutes, which is 3X faster. With providing private DNS server, it takes MGRAB 89 seconds to finish the job, which is around 34% faster than the 135 seconds taken by native ZGRAB.

4.4 VALIDATION ON DNS RESOLUTION

DNS resolution may change between a domain is processed by different parts in MGRAB. We validated the correctness of MGRAB in the aspect that such impact is tolerable. Besides comparing the success count of grabbing using MGRAB to the one using ZGRAB and get similar results, we also extensively investigated the DNS resolution stability of domains from the blacklist.

From the 100k randomly sampled URLs from the blacklist, we retrieved 99,779 unique domains. We ran ZDNS with type ALOOKUP every 20 minutes and collect the result for 5 days (120 hours).

We measured four different values:

- **percentage of domains with valid IPs**: percentage of domains with valid IPv4

	compared to **origin time point**	compared to **20 minutes ago**	compared to **origin time point** using **sliding window**	compared to **previous time point** using **sliding window**
valid IP	δ	δ	δ in next hour	δ in next hour
new IP	δ, no δ at origin time point	δ, no δ 20 minutes ago	δ in next hour, no δ at origin time point	δ, no δ in the last hour
disappear IP	no δ, δ at origin time point	no δ, δ 20 minutes ago	no δ in next one hour, δ at origin time point	δ in next hour, no δ 20 minutes ago
changed IP	δ, but the first IP from origin time point is no longer valid	δ, but the first IP from 20 minutes ago is no longer valid	δ, but the first IP from origin time point is no longer valid in next hour	δ, but the first IP from 20 minutes ago is no longer valid in next hour

Table 4.1: **IP Analysis Metrics** - The table illustrates the definitions for four different values, "valid IP", "new IP", "disappear IP" and "changed IP" in four different metrics. Note we use δ to refer to "have valid IP address".

addresses

- **percentage of domains with new IP**: percentage of domains that did not have valid IPv4 addresses in the last measurement, but have one in the next measurement
- **percentage of domains with disappear IP**: percentage of domains that had valid IPv4 addresses in the last measurement, but do not have one in the next measurement
- **percentage of domains with changed IP**: percentage of domains that changed their valid IPv4 addresses since the last measurement

We analyzed the data in four different metrics. In the first two metrics, we compared the data from every 20 minutes with 1) the data from the origin time point, 2) the data from 20 minutes ago.

In the latter two metrics, we used a sliding window to eliminate some minor instabilities in the data. We set a more rigorous rule to decide whether a domain has "no valid IP presented". Instead of looking at data from one time measurement, we defined a domain as "no valid IP presented" if and only if it has no valid IP presented in an hour (3 consecutive measurements). We also did the comparison for both the original time point and the previous time point.

The detailed description about how we decide if a domain belongs to those categories in different metrics is in Table 4.1. The measurement result are shown in Figure 4.3, 4.4, 4.5 and 4.6.

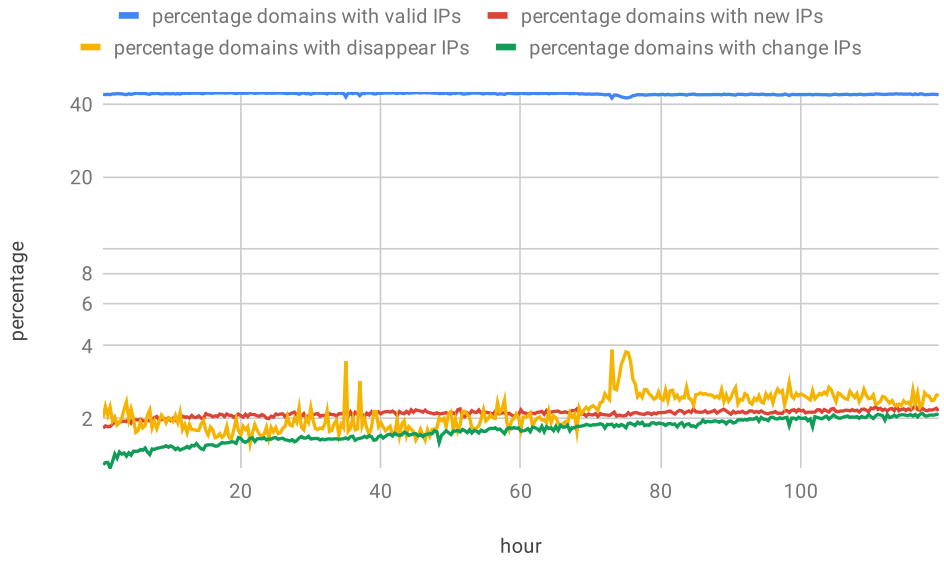

Figure 4.3: **Measurements taken when compared to origin time point** - X axis is the time in hours, and Y axis is the log percentage of different metrics compared to the total domains measured. "New IP" and "Disappear IP" fluctuate around 2% while "Change IP" slowly increased to 2% after 5 days (120 hours)

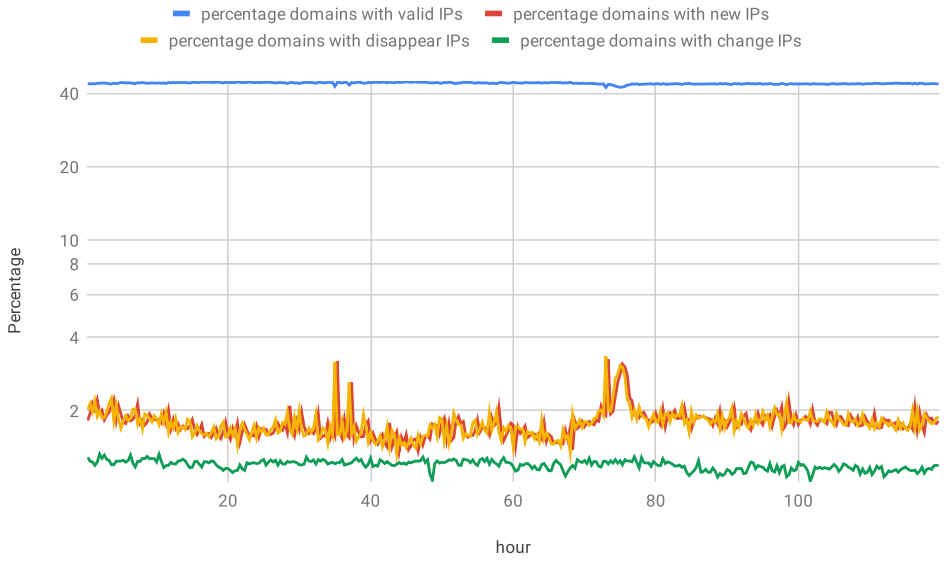

Figure 4.4: **Measurements taken when compared to previous time point (20 minutes ago)** - X axis is the time in hours, and Y axis is the log percentage of different metrics compared to the total domains measured. "Disappear IP" slowly increases while "New IP" is stable.

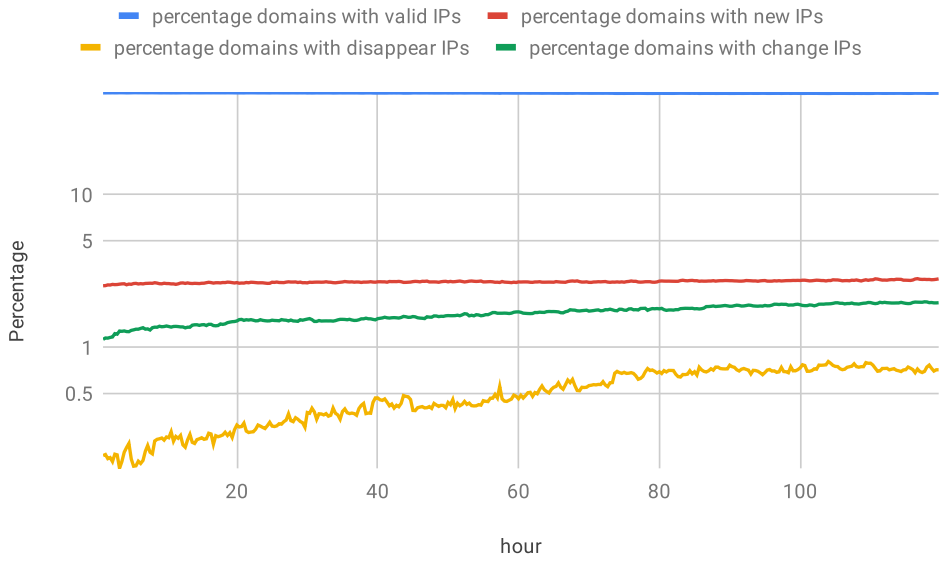

Figure 4.5: **Measurements taken when compared to origin time point using Sliding Window** - X axis is the time in hours, and Y axis is the log percentage of different metrics compared to the total domains measured. "Disappear IP" follows the trend of"New Ip" and the ratio of "change IP" is stable.

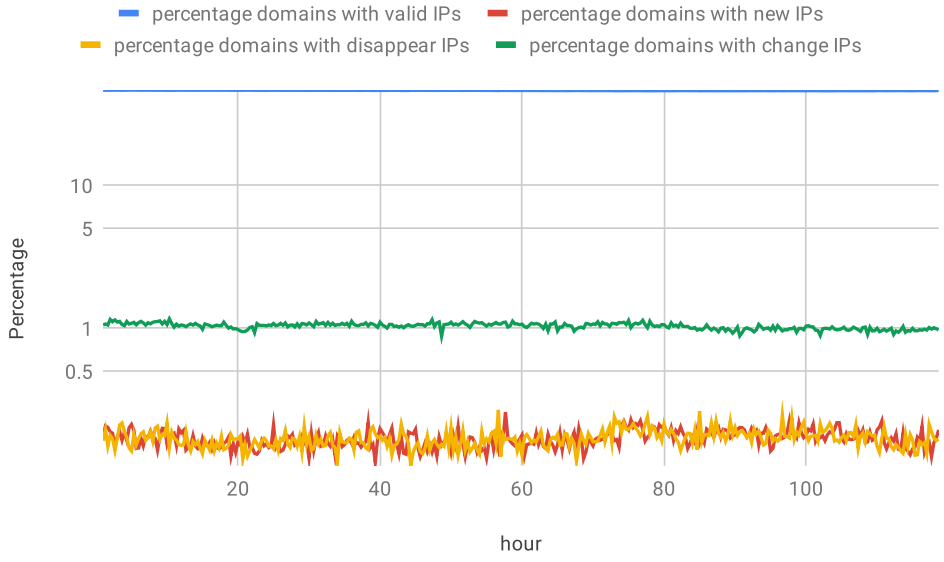

Figure 4.6: **Measurements taken when compared to previous time point (20 minutes ago) using Sliding Window** - X axis is the time in hours, and Y axis is the log percentage of different metrics compared to the total domains measured. Ratio of "new IP" and "Disappear IP" no longer share similar pattern in fluctuation and drop significantly compared to the measurement without Sliding Window.

We can obtain several takeaways from the four graphs.

First, the percentage of valid IPs is around 43%. The number is quite stable through time and different metrics.

Second, for some domains (from the graph: around 2% of the blacklist at a single point), their DNS responses could occasionally fail with no valid IPv4 addresses. But in the next queries (20 or 40 minutes later), valid responses will return. This deduction comes from the following observations:

- In Figure 4.4, we see that the trend of line representing new IP (red) follows the line representing disappear IP (yellow) closely, meaning that DNS response for those domains with disappear IPs quickly recover in the next queries, generating a line of new IPs with similar fluctuations.

- By using sliding window, we build a more rigorous criteria for "IP not presenting". We can see that for comparing to previous point, by using sliding window, the value of "percentage of domains with new IP" and "percentage of domains with disappear IP" both drop from ∼2% (in Figure 4.4) to ∼0.1 (in Figure 4.6)%.

- The lines for "new IP" in both graphs of "compared to origin time point" (Figure 4.3 and 4.5) have their starting points at ∼2% (the graph of sliding window is a little bit higher because we have a more rigorous rule for IP not presenting), but that value does not increase dramatically during the period of 5 days. The ∼2% is the initial error rate for those domains happen to have no valid DNS response in the first measurement.

Third, ∼1% of the domains change their domains frequently. For graphs compared to previous time point (Figure 4.4 and 4.6), the line of "change IP" is quite stable although the one using sliding window is more smooth. For graphs compared to origin time point (Figure 4.3 and 4.5), we see that the value of "change IP" increase slowly from 1% (after 20 minutes) to 2% (after 5 days).

To better understand the characteristics of domains behind the measurement, we generated the following CDFs (Figure 4.7 and 4.8) based on the number of changes for a domain when it is compared to the measurement result in the previous time point.

The two CDFs in Figure 4.7 and 4.8 are similar. In both graphs, the line of "new IP" is overlapped with the line of "disappeared IP" and becomes almost invisible. Due to the more rigorous criteria of "no valid IP present", the lines in graph of the one with sliding windows apparently have higher starting point and reach 100% faster. From both graphs, we can conclude that: 1. ∼97% of domains don't change their IPs frequently (not even once during five days). 2. IP disappearing and new IP appearing also become insignificant if we

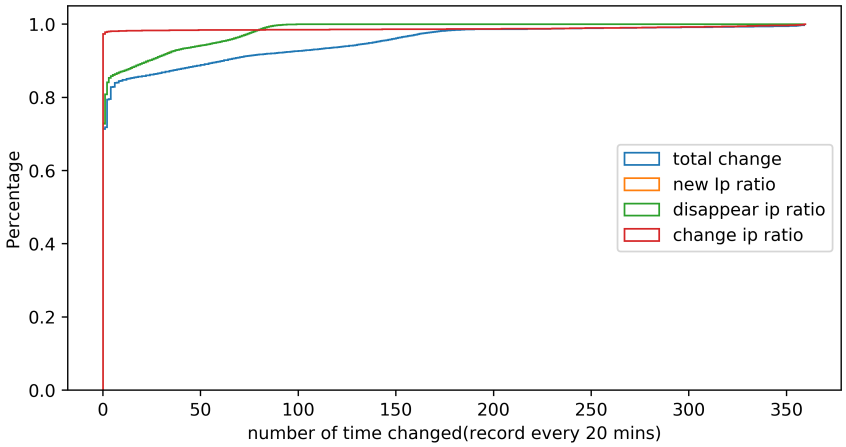

Figure 4.7: **CDF of Domain Change IP Frequency w/o Sliding Window** - "new IP ratio" (orange) is completely overlapped with "disappear IP ratio" (green) and becomes invisible.

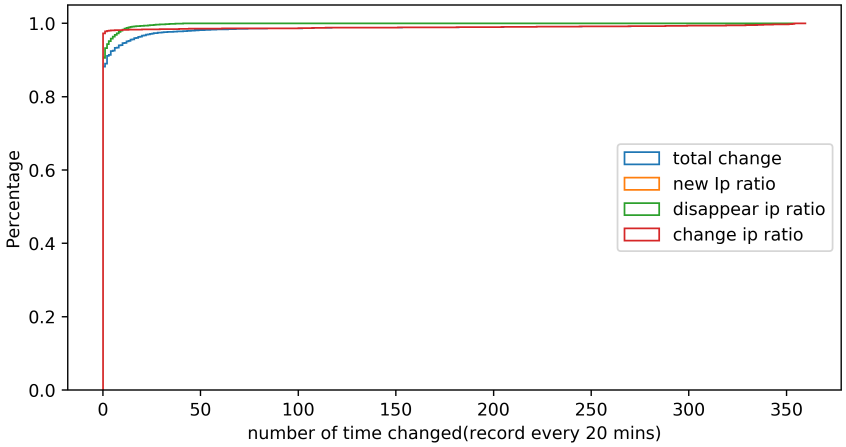

Figure 4.8: **CDF of Domain Change IP Frequency w/ Sliding Window** - "new IP ratio" (orange) is completely overlapped with "disappear IP ratio" (green) and becomes invisible. 97% of domains don't change even once in 5 days.

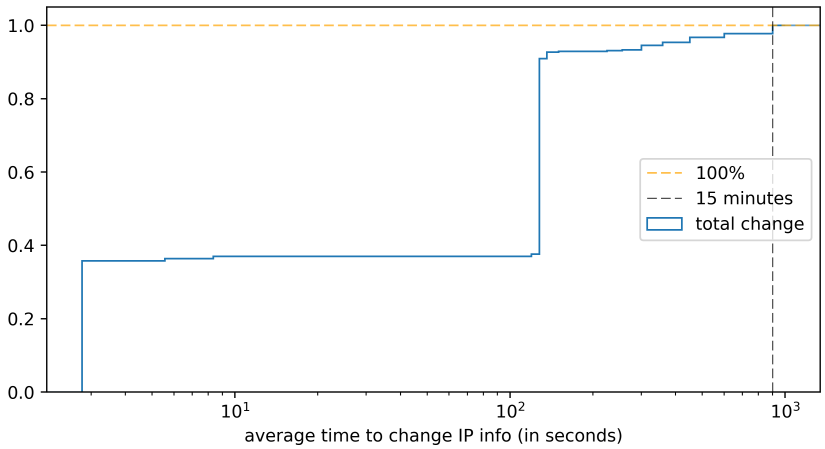

Figure 4.9: **CDF of Average Time for Domains to Change IP** - For the top 500 domains, 98% of the domains change their IP status at least once in every 15 minutes.

dont consider those "one time fail" cases. New/disappear IP ratio (green line) grows faster than change IP ratio (red line), meaning that ratio of domains frequently lose/obtain valid IP is less than the ratio of domains that frequently change their IPs.

We also took the top 500 domains from the samples that have most frequent IP status changes ("new IP", "disappear IP" and "change IP" combined, using sliding window version). We measured them in a more frequent manner (360 times in 30 minutes) and generated a CDF in Figure 4.9 about the average time of those domains that change their IP statuses.

Around 36% (36% * 500 = 180) domains change their IP status every time we measured. Around 97% (97% * 500 = 485) domains change their IP status at least once in 10 minutes. However, if we compare these numbers to the total 10K samples, we find that 0.18% of the total sampled domains change their IP status at least once every 5 seconds, 0.49% of them change their IP status at least once every 10 minutes. In conclusion, the whole sampled dataset is stable in a granularity of 10 minutes. In summary, the potential change in DNS resolution between a domain is processed by different parts in MGRAB is tolerable.

CHAPTER 5: MEASUREMENT RESULT

5.1 MEASUREMENT SUMMARY

In this section, we will analyze the success ratio of domains at different parts in the pipeline and the parked domain ratio in the successful grabbed pages. We summarize the result from one day into Figure 5.1.

5.2 MEASUREMENT RESULT FROM MGRAB

We used MGRAB to perform a daily scan on the URLs from the same blacklist for 30 days and measured the metrics recorded at different parts in the pipeline. As Figure 5.2 shows, data from most kinds of metrics are stable. In addition to Figure 5.1, we will discuss the measurement results from one day in a more detailed perspective. The ratio of unique domains to the total URLs in the blacklist is 98%, the ratio of total domains with valid IP is 42%, the ratio of unique IPs for all the domains with valid IP is 8.7%, the ratio of unique IPs with their TCP port open is 6.9%, the ratio of unique domains with their TCP port open is 42.4%, the ratio of total URLs with their TCP port open is 44.0%, the ratio of successful grabbed URLs is 39.4%. For all unique IPv4 addresses, 79% of them have their TCP port open, and for all domains with valid IPv4 addresses, 99.97% of them have their TCP port open. For all the URLs with valid IPv4 addresses and TCP ports open, 89.7% of them can be successfully grabbed. Failure cases include various possibilities such as invalid certificate or i/o timeout.

We can conclude several takeaways from the observations.

- More than half of the domains in the blacklist do not have valid IPv4 address.
- Many domains in the blacklist share the same IP address. The ratio of domains to unique IPs is around 5:1.

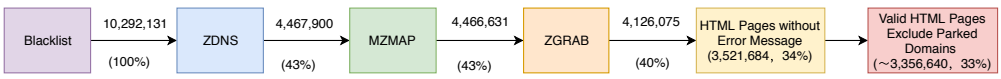

Figure 5.1: **Valid Ratio at Different Stages** - It shows the valid ratio of URLs at different stages through MGRAB and our analysis. We take data from one day as a representative, the data from other days are very close. We include both absolute count and success ratio at different stages. Overall, 33% of the total domains contain meaningful content and reach the final stage.

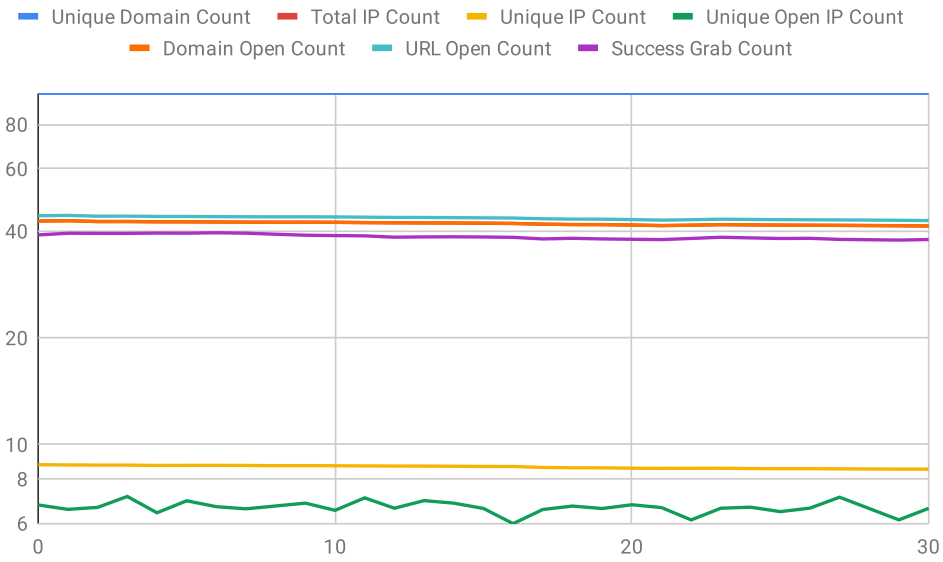

Figure 5.2: **Measurement Result from MGRAB for 31 Days** - This graph shows the success ratio of domains or IPs happened at different parts in the pipeline. Except the green line (represents the unique IP with TCP port open), all the other indexes are quite stable. Around 34% of the total URLs could be successfully grabbed. This ratio is slowly dropping, while the trend is not quite apparent.

- The count of unique IPs with TCP port open fluctuates while the count of domains with TCP port open is more stable. We assume these unstable IPs (not always open) do not have many domains related to them.
- The ratio of the domains that can be successfully grabbed is dropping slowly. However, a stronger conclusion needs data from longer time period.

5.3 PARKED DOMAIN DETECTION

In 2014, Kuhrer et al.[1] introduced methodology and result on analyzing parked domains in blacklist. We followed a similar approach, using heuristic strings and machine learning techniques to analyze the percentage of parked domains in blacklist after 5 years.

5.3.1 Ground Truth Generation

In the old paper, they split the NS hostnames into tokens and searched for terms indicating parking such as **park**, **sell**, and **expired**. They labeled NS whose hostnames match one of these terms as potential parking name servers. From domains belonging to those NS and

some other NS they detected from a passive DNS database, they extracted 47 descriptive strings, in the following referred to as identifiers (IDs). These IDs can be found in the HTTP responses of many parked domains. They use these IDs to create the parked domains dataset P that consists of 5,000 randomly chosen domains from the pDNS database they find to utilize a verified parking NS or include at least one identifier.[1]

We followed a similar approach to generate the parked domain dataset. We first looked at how those IDs from their work (referred as old ID in the following) after 5 years. Surprisingly, most of them become invalid. Only one ID is still useful and finds around 4k domains in our full grabbed data. See details in appendix A.

We look for NSs with special words (**"sell"**, **"park"**, **"expired"**) in their host names. We split the NS hostnames into tokens. If the hostname of an NS contains any special words as token, we name it as *token NS*. If its host name contains any special words as a substring, we name it as *other NS*. We verify 41 *token NS* with 5,217 unique domains. We found 1,489 *other NS* with 64,505 unique domains. We verify the top 50 *other NS* (in terms of domains belongs to it) with 59,732 unique domains, which accounts for 93% of the total domains belongs to all *other NS*. In total, we verified 94 parked NS with 64,350 unique domains.

We observed that many old IDs are URLs. We extracted the most frequent appearing URLs from the 64,350 unique domains and verified 21 of them as our new IDs. We also manually extracted 9 common pattern of domains belong to several large parked NS. These new IDs help us found 112,516 unique domains (see Appendix B), 66.1% of which are not covered from the discovered parked NS.

By examining the additional domains, we also discovered 4 new parked NS. Combining the results found by *token NS*, *other NS*, useful old ID, new IDs and the new NS, we were able to identify 139,646 parked domains, which is 3.38% of the 4,126,075 unique domains that we successfully grabbed for that day.

The paper[1] uses 5,000 parked domains and the top 5,000 domains from Alexa for training dataset. We used the same source for non-parked dataset after removing all the parked domains discovered by our IDs. We increased the number of both parked and non parked domains to 10K for the purpose of improving the model performance.

5.3.2 Feature Selection

We also made some changes to the features used for the model. We first removed some features that cannot be easily retrieved from our collected data (e.g. whether the connection to a random subdomain could be successful) or became quite useless (e.g. the appearance of <**frame**> tag in HTML pages become very unusual, both in parked and non parked

domains). We also changed the way to represent <**robot**> value specified in <**meta**>. Instead of mapping a binary value like **index** and **follow** to a bit and cast all the binary values into a 32 bits integer, we believe having different representation of each binary value is more powerful considering some integers (binary value combinations) may be rare in the dataset. We also evaluated the tag <**googlebot**> together with <**robot**> as it is also a quite popular tag servers similar functionality as the <**robot**> tag. Finally, we also observed that the HTML page length of parked domain tends to be short as they do not need to deliver much information or complete complex functionality. We also added the length of the of the whole HTML page as an additional feature. We used 7 features in total:

- length of the whole HTML page
- ratio of human-readable text in relative to the overall length in returned web content. Many parked domains deliver less human-readable text compared to non parked domains.
- ratio of JavaScript code in relative to the overall length in returned web content. Many parked domains use JavaScript code to display advertisements while the displayed human-readable text is relatively low.
- the average length of <**anchor**> tags. Some parked domains rather use long attributes in the referral <**anchor**> tags and result in longer average length.
- allow **index** in <**robot**> or <**googlebot**>. Parking providers monetize the domains and are interested in promoting their domains, thus permitting indexing by search engines.
- allow **follow** in <**robot**> or <**googlebot**>. Similar to the fourth feature.
- allow **archive** in <**robot**> or <**googlebot**>.Similar to the fourth feature.

5.3.3 Training Phase

Instead of SVM, random forest achieved the best performance on our dataset. We evaluated the feature set with a 10-fold cross validation and achieved an average detection rate of 99.7% correctly classified domains while the false positive (FP) rate is 0.06% and the false negative (FN) rate is 0.22%. See the feature importance in table 5.1: We manually examined the 5 FP and 9 FN cases. For false positive cases where non parked domains are categorized as parked, one of them is with little human-readable text("egexa.com", Figure

Not Secure | view-source:egexa.com

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, height=device-height, initial-scale=1.0">
<meta name="apple-mobile-web-app-title" content="egexa">
<meta name="application-name" content="egexa">
<meta name="msapplication-TileColor" content="#ffffff">
<meta name="theme-color" content="#ffffff">
<title>eGexa</title>
<link href="//egcdn.egexa.net/sc/aGjzrbGAGhGXrPucGLbPGPaOPBGg-v2.css" rel="stylesheet" type="text/css">
</head>
<body>
<div class="main">
<h1><span class="e">e</span><span class="g">G</span><span class="e">exa</span></h1>
</div>
<div class="cp">&copy; 2018 eGexa</div>
<script async src="https://www.googletagmanager.com/gtag/js?id=UA-3676303-1"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'UA-3676303-1');
</script>
</body>
</html>
```

Figure 5.3: **False Positive Example 1** - This site shows only one word and contains only 27 lines of HTML source code. It has small value for both raw page length and text ratio, which makes its structure very similar to a parked domain.

Not Secure | view-source:mycloud.com/#/

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My Cloud</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-scalable=no">
    <link rel="stylesheet" href="./stylesheets/index.css" />
    <link rel="shortcut icon" href="./favicon.ico" />
    <link rel="icon" href="./favicon.ico" />
    <script>
      var utag_data = {};
    </script>
  </head>
  <body>
    <script type="text/javascript">
      (function(a,b,c,d){
      a='//tags.tiqcdn.com/utag/wd/main/prod/utag.js';
      b=document;c='script';d=b.createElement(c);d.src=a;d.type='text/java'+c;d.async=true;
      a=b.getElementsByTagName(c)[0];a.parentNode.insertBefore(d,a);
      })();
    </script>
    <div id='root'></div>
    <script type="text/javascript" src="./javascripts/index.js"></script>
  </body>
</html>
```

use JavaScript code from elsewhere

Figure 5.4: **False Positive Example 2** - The HTML source code of this site is also short as the majority of the code is imported from elsewhere. It also relies heavily on JavaScript code to deliver human-readable text. Both features, raw page length and human readable text ratio, increase the possibility of being identified as parked by the model.

rawPageLen	jsCodeTotalRatio	aTagLen	textTotalRatio	index	follow	archive
0.425652	0.227170	0.179904	0.155608	0.007868	0.002329	001470

Table 5.1: **Feature Importance** - The table illustrates the contribution of different features to the total decision.

5.3). The other four domains tend to embed human-readable text in anchor tag or JavaScript code and make these content identified as non human-readable text, thus lowering the ratio of human-readable text and raising the ratio of JavaScript code in some cases. Some sites may even import the JavaScript code from elsewhere (like "mycloud.com", Figure 5.4), thus decreasing the total page length. For false negative cases where parked domains are categorized as non parked, the causes could be various extreme feature values. For example, some sites have long page length because of large amount of text/code for advertisement from the parking providers, or because they still show some of their original information while expired. Some parked domains may just do not use JavaScript code at all and make the JavaScript code ratio 0. The long raw page length, large human-readable text ratio and small JavaScript code ratio are common features for non-parked popular websites.

5.3.4 Applying to All Grabbed Data

We applied the model to the data collected for one day, where there are 4,149,558 successful grabbed HTTP pages with 4,126,075 unique domains. The machine learning model decides 759,156 (18.4%) of the unique domains are parked. We found that 586,600 (14.2%) cases are returned with " status_code" of "429 Too Many Requests". We believe the error is not caused by our IP address reaching the rate limit as we only run the scan once a day. These domains are not working properly. We also found that 17,791 (0.043%) domains are showing message "502 Bad Gateway". Finally we have 154,765 (3.75%) domains labeled as parked by the model, 35,523 (0.86%) of them are not identified by the IDs or belong to any special NS. We manually inspected a sample of 20 domains belong to this case. 15 of them are parked, 1 of them is no longer available, and 2 of them are gambling and lottery websites.

The false positive rate and false negative rate of applying the model to all the collected data are higher. We believe that the reason is caused by the difference between training data and the real data. In training dataset, we only used domains match our IDs as parked. There are many domains belong to parked NS but not present typical parked behavior in the HTML pages grabbed. Many of them contain error messages like "Error. Page cannot be displayed." in the grabbed data but showed parked page when we visited them through a browser a month later. These domains are the major contributors to the false negative cases.

We used popular domains from top Alexa for non parked data during the training phase. There are many dissimilarities between these popular websites and the non parked domains in the blacklist. Blacklist contains various kinds of domains. Phishing websites may deliver similar contents compared to popular websites. But there are also more websites regarding gambling, lottery and pornography from the blacklist. These websites tend to be simple, deliver little human-readable text, and embed much JavaScript code. All these features are closer to parked domains rather than non parked popular domains from Alexa top. These domains are the major contributors for the false positive cases.

5.3.5 Conclusion about Parked Domain Ratio

If we use 75% as the true positive rate in the set of domains newly labeled as parked by the model, the total ratio of parked domains is 3.38% (ratio of parked domains identified in the data generating stage) + 0.86% (ratio of parked domains newly labeled by the model) * 75% (true positive rate in assumption) = 4.0%. This number also include some domains that belong to parked NS but not present park behaviors during the time we grabbed them. In general, the parked domain ratio is much smaller than five years ago. In the experiment in 2014, they found 9% of the total grabbed domains to be parked based on identifiers and used the machine learning model to found 17% of the total grabbed domains to be parked. The quality of the blacklist is hugely improved in the aspect of excluding parked domains.

CHAPTER 6: LIMITATIONS

There are several limitations regarding MGRAB and our analysis. As for MGRAB, it inherits limitations from the software tools it uses. For $\sim$2% of domains in the blacklist, DNS requests may occasionally fail with no valid IPv4 addresses. As ZMAP is finger-printable, some websites may implement mechanism to remain hidden from ZMAP scan. Some malicious websites also do "cloaking" which presents different content to the ZGRAB scan compared to visiting them through browsers. All of these may introduce minor inaccuracies in the measurement.

As for parked domain detection, the training dataset we used for the machine learning model is not perfect. As we have mentioned in the previous section, the popular websites which are used for non-parked dataset have many dissimilarities compared to the non-parked domains in blacklist. A solution to this is using websites from blacklist for non-parked dataset, but requires large amount of labor.

CHAPTER 7: FUTURE WORK

Some work could be improved and extended. First, MGRAB only supports IPv4 at this time. It is possible and helpful to enable MGRAB to support IPv6, which will become more popular in the future. Analysis work on measurement data could also be extended. We could analyze the DNS trace of all domains collected, like extracting the name servers with most domains in the blacklist. With data from a longer time period, it is possible to do more analysis and obtain more deductions.

As MGRAB is a comprehensive and efficient banner grabber, researchers can use it to investigate how some malicious websites attempt to avoid detection. Researchers can repeat experiments with controlled variable, e.g. user agents, IP, JavaScript Check and browser fingerprinting, in order to find how changing these variables affect the status/content of domains collected.

Finally, by excluding unavailable domains gathered at different stages in MGRAB and parked domains detected by our methodology, researchers can build a curative blacklist with higher quality.

CHAPTER 8: CONCLUSION

In this work, we modernized an existing approach to identify offline domains and parked domains in blacklists. We built and open-sourced an efficient blacklist filter, MGRAB, that can filter domains and exclude those that are unresolvable, offline, or inaccessible. Using MGRAB, we found that only 43% of total domains have valid IPv4 address and only 40% of the total domains can be successfully grabbed. We also proposed an updated mechanism to detect parked domains using new heuristic strings and new machine learning models. Using our updated classifier, we found that around 4% of the successfully grabbed domains are parked. Overall, only 33% of the total domains contain meaningful content. Researchers can use MGRAB and the parked domain detection methodology to generate a pre-filter of blacklists for security research. This is a necessary first step towards a range of future security research, including a holistic view of web cloaking or other behavior of malicious websites.

REFERENCES

[1] M. Kührer, C. Rossow, and T. Holz, "Paint it black: Evaluating the effectiveness of malware blacklists," in *Research in Attacks, Intrusions and Defenses*, A. Stavrou, H. Bos, and G. Portokalidis, Eds. Cham: Springer International Publishing, 2014, pp. 1–21.

[2] "Making the worlds information safely accessible." 2019. [Online]. Available: https://safebrowsing.google.com/

[3] C. Kolbitsch, B. Livshits, B. Zorn, and C. Seifert, "Rozzle: De-cloaking internet malware," in *2012 IEEE Symposium on Security and Privacy*, May 2012, pp. 443–457.

[4] "How the domain name system (dns) works," 2019. [Online]. Available: https://www.verisign.com/en_US/website-presence/online/how-dns-works/index.xhtml

[5] "zmap/zdns," 2019. [Online]. Available: https://github.com/zmap/zdns

[6] "The zmap project," 2019. [Online]. Available: https://zmap.io/

[7] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications," in *USENIX Security Symposium*, 2013.

[8] "Banner grabbing," 2019. [Online]. Available: https://en.wikipedia.org/wiki/Banner_grabbing

[9] "zmap/zgrab," 2019. [Online]. Available: https://github.com/zmap/zgrab

[10] "Domain parking - wikipedia," 2019. [Online]. Available: https://en.wikipedia.org/wiki/Domain_parking

[11] S. Sheng, B. Wardman, G. Warner, L. Cranor, J. Hong, and C. Zhang, "An empirical analysis of phishing blacklists," 2009.

[12] T.-F. Yen and M. K. Reiter, "Traffic aggregation for malware detection," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2008, pp. 207–227.

[13] D. Canali, M. Cova, G. Vigna, and C. Kruegel, "Prophiler: A fast filter for the large-scale detection of malicious web pages," in *Proceedings of the 20th International Conference on World Wide Web*, ser. WWW '11. New York, NY, USA: ACM, 2011. [Online]. Available: http://doi.acm.org/10.1145/1963405.1963436 pp. 197–206.

[14] B. Eshete, A. Villafiorita, and K. Weldemariam, "Binspect: Holistic analysis and detection of malicious web pages," in *Security and Privacy in Communication Networks*, A. D. Keromytis and R. Di Pietro, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 149–166.

[15] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Learning to detect malicious urls," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 30:1–30:24, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1961189.1961202

[16] X. Hu, M. Knysz, and K. G. Shin, "Rb-seeker: Auto-detection of redirection botnets," in *In Network & Distributed System Security Symposium*, 2009.

[17] Y. Zhauniarovich, I. Khalil, T. Yu, and M. Dacier, "A survey on malicious domains detection through dns data analysis," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, p. 67, 2018.

[18] J. Lee, J. Kwon, H. Shin, and H. Lee, "Tracking multiple c amp;c botnets by analyzing dns traffic," in *2010 6th IEEE Workshop on Secure Network Protocols*, Oct 2010, pp. 67–72.

[19] J. Lee and H. Lee, "Gmad: Graph-based malware activity detection by dns traffic analysis," *Computer Communications*, vol. 49, pp. 33–47, 2014.

[20] A. Oprea, Z. Li, T.-F. Yen, S. H. Chin, and S. Alrwais, "Detection of early-stage enterprise infection by mining large-scale log data," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2015, pp. 45–56.

[21] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Y. Zhao, "Detecting and characterizing social spam campaigns," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 35–47.

[22] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and evaluation of a real-time url spam filtering service," in *2011 IEEE Symposium on Security and Privacy*, May 2011, pp. 447–462.

[23] S. Sinha, M. Bailey, and F. Jahanian, "Shades of grey: On the effectiveness of reputation-based blacklists," in *2008 3rd International Conference on Malicious and Unwanted Software (MALWARE)*, Oct 2008, pp. 57–64.

[24] C. Rossow, T. Czerwinski, C. J. Dietrich, and N. Pohlmann, "Detecting gray in black and white." MIT Spam Conference, 2010.

[25] C. J. Dietrich and C. Rossow, "Empirical research on ip blacklisting." in *CEAS*, 2008.

[26] "orcaman/concurrent-map," 2019. [Online]. Available: https://github.com/orcaman/concurrent-map

[27] "Headless chromium," 2019. [Online]. Available: https://chromium.googlesource.com/chromium/src/+/lkgr/headless/README.md

[28] "telnet," 2019. [Online]. Available: https://telent.com/

[29] "redis," 2019. [Online]. Available: https://redis.io/

[30] "Majestic million csv now free for all, daily." 2019. [Online]. Available: https://blog.majestic.com/development/majestic-million-csv-daily/

APPENDIX A: RESULT OF OLD IDENTIFIERS

Identifiers that find domain match:

applyFrameKiller(); *find 3963 domains match*

<frame src="http://adomino.com/p.php?domain= *find 4 domains match*

landingparent *find 2 domains match*

All the other identifiers find no domain match:

<framesrc="http:/ \/ \ww[09]1,2 \.*(regular expression)*

__gpup \(http:/ \/ \ww[09]1,2 \.*(regular expression)*

'resultsPageBaseUrl':'http:/ \/ \ww[09]1,2 \.*(regular expression)*

\$.ajax \(\url:'http: \/ \/ww[09]1,2 \.*(regular expression)*

http: \/ \/05000[09]1.voodoo.com*(regular expression)*

__elm.setAttribute("src","http://contextual.media.net/dmp

type="text/javascript">document.write(popularCategories);</script><a

'resultsPageBaseUrl': 'http://chargedodia.com'+pq,

/common/roar/landing/rpos/">

s reserved with Domain Discover

<ahref="/inquiry.html?Query=

www.domainpower.com/marketplace/domainfor

<framesrc="http://www.domainleasing.com/"

<framesrc="http://dsparking.com/?o

"0"scrolling=nosrc="http://landing.domainsponsor.com

&domainname=refererdetect'

https://www.bodis.com/market/checkout?domain=

https://mcc.godaddy.com/parked/park.aspx

https://mcc.godaddy.com/DomainInterest

https://mcc.securepaynet.net/parked/park.aspx

This page provided to the domain owner free by Sedo

href="http://www.domainpool.com/domains/

http://hus.parkingspa.com?domain=

http://hus.parkingspa.com/?cid=

http://fwdservice.com/main.php?dmn=

http://www.searchnut.com/?domain=

http://www.trafficzposter.com/inquiry.html?Query=

Privacy Policy
href=”http://domainnamesales.com/”>DomainNameSales.com
src=“http://return.bs.domainnamesales.com/
 <a href=“http://www.buydomains.com
src=”http://return.uk.domainnamesales.com/
var popularCategoriesHardCoded = newArray();
<LI class=style4> <a class=style4 href=”dtzsch/
<a class=links href=“dtzsch/
Privacy Policy
href=“/xflp/_layout_/layout
<frame id=“sub1” src=“bh.php?dm=
uri=“/checkMouse.php”+”?idssid=
extra_pages/parkingdomainprices.png”
http://website.ws/construct.dhtml?host=
This page is provided courtesy of <a href=“http://www.godaddy.com?
href=“http://www.godaddy.com/domains/search.aspx?domainToCheck=
<a href=“http://fwdservice.com/domaininquiry.php?d=

APPENDIX B: RESULT OF NEW IDENTIFIERS

http://parkingcrew.net *find 68583*
http://parking.parklogic.com *find 10512*
http://img.sedoparking.com *find 17231*
http://sedoparking.com *find 16976*
https://www.sedo.com *find 8087*
http://www.parkingcrew.net *find 51615*
http://i.cdnpark.com *find 51547*
http://plp.parklogic.com *find 115*
https://www.afternic.com *find 3246*
http://c.parkingcrew.net *find 3344*
https://www.domains4bitcoins.com *find 1743*
https://domains4bitcoins.help *find 1743*
https://my.domains4bitcoins.com *find 1743*
https://www.reg.ru *find 2998*
http://whois.co *find 1354*
https://www.mydomaincontact.com *find 1251*
https://parking.reg.ru *find 1862*
http://www.sedo.com *find 657*
https://sedo.com *find 1481*
http://www.afternic.com *find 717*
https://static.uniregistry.com/static/assets/dist/css/uniregistry.css *find 17*
saScript.src = location.protocol + "//pxlgnpgecom-a.akamaihd.net/javascripts/browserfp.min.js ?templateId=10"; *find 54*
Terms of Use *find 2094*
href="http://www.go.co/register/?src=SuspensionPage&searchurl=*find 1650*
If this is your domain name you must renew it immediately before it is deleted and permanently removed from your account. *find 584*
href="//www.namebright.com" *find 883*
adblockkey="MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAKX74ixpzVyXbJprcLfbH4psP4 +L2entqri0lzh6pkAaXLPIcclv6DQBeJJjGFWrBIF6QMyFwXT5CCRyjS2penECAwEAAQ== *find 10969*

https://www.namesilo.com/parking/assets/style.css *find 2395*
The Sponsored Listings displayed above are served automatically by a third party. Neither the service provider nor the domain owner maintain any relationship with the advertisers. *find 15759*
href="/gdforsale/static/css/main.8bd97846.css" *find 2618*
https://www.godaddy.com/offers/domains?isc=GPPTCOM&utm_medium=parkedpages &utm_source=godaddy *find 3078*