Research Report : DNSg

M. Fields †, K. Miller †, Y. Shah ‡, V. Smirnov †, and S. Yawalkar ‡

†Department of Computer Science
‡Department of Electrical and Computer Engineering
Georgia Institute of Technology

***Abstract*—In this paper we introduce both capd, a tool designed for scalable collection and analysis of netflow data, and our results utilizing it to investigate the implicit DNS landscape of the internet. By gathering DNS data from visits to sites taken from the Alexa Top 1M, we are able to construct a bipartite graph between the primary seed domains and secondary implicit domains and IPs the browser connects to on each visit. From this, we build a model for expected DNS behavior based on the site's Alexa rank, and later incorporate publicly available domain and IP blacklists to identify characteristics of low-ranking or malicious sites.**

I. Introduction

The Domain Name System (DNS) [5], [6] maps domain names to IP addresses, and provides a core service to applications on the World Wide Web (WWW). Figure 1 shows the overview of a DNS query cycle. Internet based attacks often leverage the use DNS systems to mount attacks against unsuspecting users. For example, spywares exfiltrate user information to certain drop sites by using anonymously registered domains. Botnets make use of disposable domains to host malicious websites such as advertising content and use the short life of the domains to avoid blacklisting. As a result, adversaries take advantage of DNS agility in order to blend in their malicious activities with that of benign DNS traffic. By understanding the nature of benign DNS traffic, we can then detect the nature of non-benign traffic and use that to track this suspicious activities.

To this effect it becomes essential to also leverage the use of DNS systems and the information obtained by passively analyze domain related DNS traffic to mitigate such attacks from occurring. In this paper we focusing on the HTTP traffic generated when attempting access to a domain. We will be observing the request-response sequences generated when accessing a website using a browser. More specifically the behavior and nature of the secondary request calls generated by a website during its rendering. Figure 2 depicts the list of secondary domains generated when querying yahoo.com .
We endeavor to leverage the information obtained by monitoring, analyzing and categorizing these secondary calls, and use this information to generate features to be utilized by our analysis engine for classifying a seed domain and it's associated implicit secondary level domains (SLD) as having benign or suspicious behavior. Some of the main features are:

- number of Internet Protocol (IP) Addresses that secondary domains resolve to
- number of Content Delivery Networks (CDN) per seed domain
- variability of the geographical location of the secondary domains

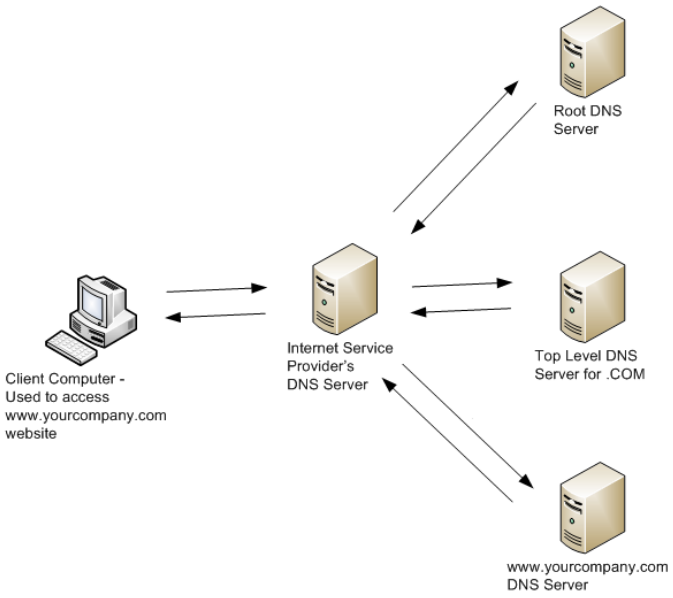

Fig. 1. Example of Domain Name Resolution process [11]

These features are leveraged to understand the behavior of implicit calls made in the process of accessing an explicit (seed) domain, as a basis for a system that is able to model the DNS behavior based on the site's Alexa rank, and later incorporate publicly available domain and IP blacklists into our training model to identify malicious traffic based on the characteristics of the modeled behavior. We attempt to cluster secondary providers based on their purpose and utilization at different tiers of the Alexa 1M, in hope that the knowledge gained from this endeavor can be ultimately utilized in the creation of a system that can be used to dynamically block secondary calls from being made if their behavior is determined to be maliciousness in nature.

In order to collect the necessary pDNS data that would be required for this project, we would need to build a web crawler. A Web crawler is a program that is used to scrape the content of a web page and create tags for features such

DNSg, DNS group, is the name of the system that has been built
Authors alphabetically arranged

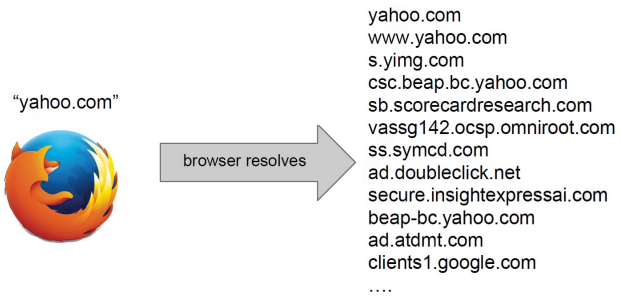

Fig. 2. Example of secondary domain calls for yahoo.com

as hyperlinks, HTML tags, etc so as to create an index of the data. By allowing the crawler to crawl a specific list of websites, such as the domains listed in Alexa 1M, we are able to record a snapshot of those websites. This allows us to perform analysis of these websites offline, and without the possibility of comparisons of websites being affected over time, as the information of the web page have been stored for a given time. However, in order to have accurate collection of website data, we have to ensure that we crawl the list of web pages within a reasonable amount of time to avoid time based changes of the website from corrupting the dataset.

Crawling our list of seed domains in its entirety the data in order to obtain the dataset is a time consuming task and resources intensive task if done serially [7]. We have extended the concept used in [8] so as to run parallel instances of web crawlers in order to reduce the net time needed to crawl all seed domains. As all the domains would be crawled with a smaller time variation, it would reduce the data variability over time leading to more accurate results.

II. Background and Related Work

DNS is the protocol that resolves a domain name, like www.foo.com, to its corresponding IP address, for example 192.11.12.14. To resolve a new domain whose resolution is not present in cache, a host will request the local recursive DNS server (RDNS). The recursive server will iteratively discover the Authoritative Name Server (ANS) who is responsible for each zone. This leads to the mapping of the requested domain name to its current IP addresses.

Zdrnja et al. in their paper "Passive Monitoring of DNS Anomalies" [4] discuss how pDNS data can be used as a basis for acquiring security features of a certain domain name. They propose the idea of using these features to build a system that can be used to give reputations for a certain domain, but do not formally implement such a system. Antonakakis et al. have built two such systems [1], [2] that make use of the pDNS data to build DNS reputation systems. The two systems monitor the DNS resolutions at different levels to extract features of the domains in order to classify a domain name as benign and malicious.

Hao et al. [3] published a report on DNS lookup patterns measured from the .com TLD servers. Their work was able to show that the resolution patterns are different for malicious and legitimate domains. In complementary to these works, our makes use of pDNS data in order to model the nature of DNS traffic of the top, middle and lower ranks of Alexa 1M, and determine the variation among these three classes of domains. This system can then be used to determine the expected rank of a certain domain, thus flagging any domains as suspicious if its Alexa rank has a large deviation from their DNS behavior.

For the webcrawler, Wu et al. [8] describe a system that runs webcrawlers in parallel, in a cluster in order to enhance the speed of crawling as well improve system performance. We have taken inspiration from the performance improvement observed by this work in order to design our novel webcrawling system, in which multiple instances of crawling take place simultaneously. This system allows us to have the benefits of the speed up and prevent a single point of failure that was highlighted by [8] without the synchronization and co-ordination required by a parallel system architecture. The details of this system have been described later in the paper, under *C*APD.

III. System Overview

To obtain the relevant data for our study, it is necessary to query the Alexa 1M list in its entirety multiple times. Completing this task once for all 1 million domains sequentially using a common browser such as Google Chrome or Mozilla Firefox would take approximately 2 months using an 4 core system with a dedicated broadband connection.

This is not feasible because to get the bare minimum points for comparison the system would have to be live for at least 6 months, and if the domains are run in parallel, an additional computational overhead would be required to parse and separate the traffic for each seed domain, thus also not feasible. To mitigate these limitations and meet the demands of scalability, the domain queries or scrapes were carried out in parallel using isolated Docker containers managed by our CAPD daemon. The System Architecture of DNSg can be seen in Figure 3.

A. *CAPD*

This daemon is the core process that is used to run the Docker container and manage the underlying processes, i.e. scraper and logger, that are to be run within the container. Capd is designed to be extensible and accommodate a range of implementations at each level. The lifecycle of capd is as follows:

i) capd initializes the environment and loads the Alexa 1M list.

ii) A docker container is spun around the traffic logger, which is used to capture the traffic specific to seed domains being queried.

iii) Docker containers are spun around each seed domain currently being queried, and the containers are assigned different MAC addresses for easy segregation of traffic from the logger generated pcap.

iv) scraper visits seed domain and exits container.

v) capd destroys the container and load the next domain.

vi) after all the domains have been crawled and no new task

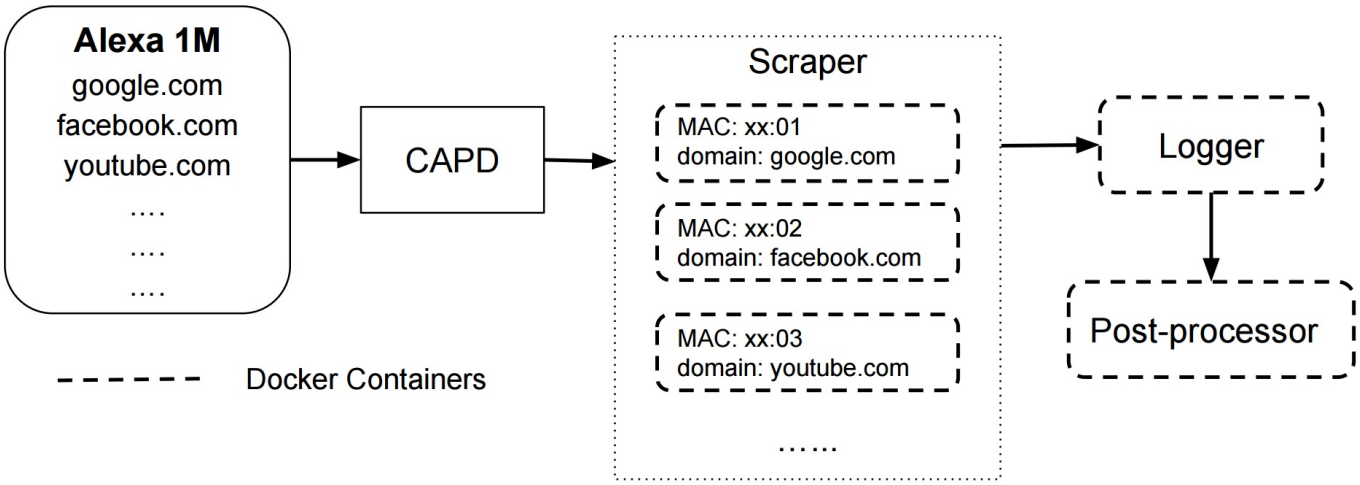

Fig. 3. System Architecture of DNSg

are initiated by the logger, the logger exits and the container is destroyed.
Best Results are obtained when (Number of CPUs – 1) container threads are used, i.e, 7 parallel instances for a octa-core processor.

B. *Scraper*

The required data for this system is not the content of the domain being queried but the underlying network traffic associated with each query. For our data collection needs 2 different types of scrapers were tested for data collection.

1) Web-crawler: Scrapy is a python based web-crawler library used for quick and efficient crawling and scraping of web content. Although fast, it does not generate the underlying traffic for the queried domain and only returns the content of the seed domain. Thus was deemed unfeasible as data collection source. We tested other crawlers like GRUB and Sphider, but they too displayed results similar to Scrapy. Due to this, the choice of using web-crawlers was deemed unfit.

2) Headless Browser: Due of the limitations presented by the use of web-crawlers, we switched to headless browsing for generating the network traffic. The choice of a headless browser to that of a full browser is due to the smaller computational and processing overhead incurred as opposed to a full browser with higher execution latency. Initially we tested the Mozilla Firefox headless browser which visualizes an X-display and runs a full browser within. This generates the DNS resolution needed but it proved to be prohibitively slow. Therefore, a switch was made to the PhantomJS headless browser. This proved to be a much lighter choice for a scraper and provided the necessary resolutions needed to finish the crawling efficiently (the secondary domain resolution life cycle as captured by PhantomJS is depicted in figure 4). The use of headless browsers enabled us to crawl approximately 30,000 domains in a day using a 6 core processor with only 4 cores dedicated to crawling. But the data demands still overwhelmed our system forcing us to limit the number of crawled domains to only 150,000, which was successfully queried in its entirety in approximately 5 days.

C. *Logger*

The logger maintains and monitors any and all request timeouts that may occur during the querying process due to unavailable system or network resources and log them so that these missed domains may be re-queried after the completion of all tasks. The logger daemon also collects and records all network traffic using tcpdump and stores it into pcap files, for the lifetime of the container. The logger also ensures the pcap rollovers (splitting captured traffic into different pcaps due imposed max size limit of the pcap file) and monitors traffic around the pcap edges to ensure that no traffic is dropped or missed.

D. *Post Processor*

The post-processor parses out the pcap and log files as necessary, and persists the record to Postgre. We are storing our data in PostgreSQL database. The data is stored as tables, where seed domains are connected to tables of secondary domains. We are using the combination of date and seed domain as our keys for each entry. Though a different database system like Neo4J might be used due to the graphical nature of the dataset.

From above it is evident that the Scraper, Logger and Post-Processor are sub daemon to the primary capd daemon. By the design of capd, we can extend its function to run additional processes for the collection of new information for example, IP geolocation, dig/nslookup, BGP AS broadcasting the IP, etc in parallel to the current implementation. Thus ensuring the scalability of the system and making it more versatile, by allowing for the inclusion and integration of other processes into the existing system.

IV. Methodology

The pcap files obtained from the logger is parsed through using 'dpkt', a python based library, and the resource records

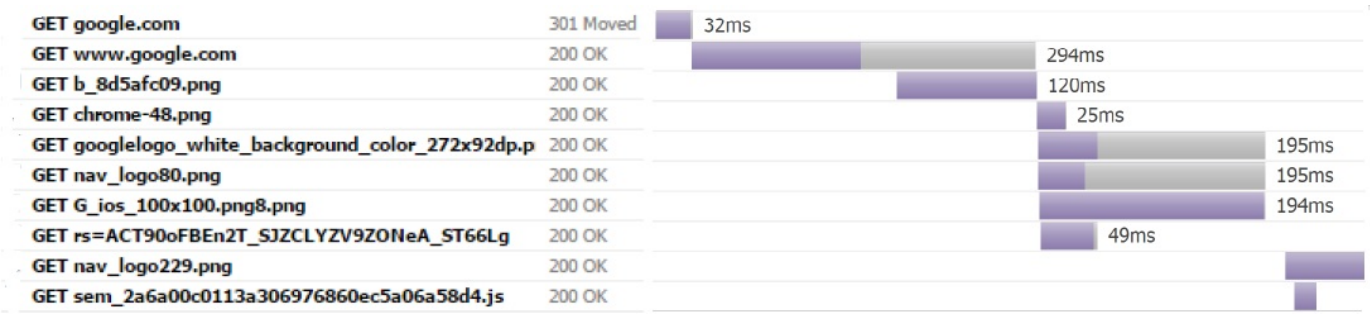

Fig. 4. Secondary domain resolution life cycle

extracted. This information includes requested domains, their corresponding cnames, resolved IPv4/IPv6 addresses and their response codes. We additionally extract the Alexa rank of each seed domain by decoding the virtual MAC address. This data is stored in a PostgreSQL database for relatively easy access.

The features used by our system are then extracted from the raw resource record data through SQL aggregation queries and geo-ip database look-ups. The SQL aggregation queries were relatively slow at first, but after proper indexing we were able to achieve reasonable response times.

A. *Features Used*

In this section we will elaborate on the features that we have used for our classifier. It is to be noted that we have used features from prior work predominantly [1]–[3]. The following describes each feature with justification.

1) **IP based features:**
   a) *Total IPv4/IPv6 addresses resolved for secondary domain* (Label: n_ipv4 & n_ipv6)
   This feature maintains a count of the number of unique IPv4/IPv6 addresses that all the CDNs resolve to. This feature gives an insight into the scale of operations of a certain domain, as a popular domain will have multiple CDNs across the world, thereby increase the number of uniquely resolved IP addresses. On the other hand, domains that are lower ranked should have a lower number of unique IPv4 resolutions and even fewer IPv6 resolutions. This number can thus be used to segregate the resolutions of the higher and lower ranked Alexa domains.
   b) *Average IPv4/IPv6 address resolutions per query* (Label: avg_n_ipv4_per_query & avg_n_ipv6_per_query)
   This feature gives the average number of IPv4/IPv6 addresses a request returns. This feature provides further insight to the size of the infrastructure of the requested CDN. We believe that domains higher up in the top 1 million will resolve CDNs with a larger infrastructure and more available IP addresses space and have a more significant integration with IPv6 address space when compared to lowest ranked Alexa domains.
2) **Domain Based Features:**
   a) *Number of CDNs per seed domain* (Label: n_distinct_slds)
   The number of CDNs per seed domain is the number of unique secondary domains that were observed for the given seed domain. This feature deals more with the genre a domain can be classified into, rather than its Alexa's rank. Websites with similar function, like two news dissemination websites, will contain multiple links to other articles and advertisements, both within its Authoritative Name Server (AuthNS) as well as outside, and thus display similar network characteristics irrespective of their Alexa rank. Conversely websites having similar ranks may not contain same features due to different nature of the websites. Thus this feature is used to cluster domains on the basis of their behavior and nature rather than their rank.
   b) *Number of change of cname records* (Label: n_cname)
   This feature gives the total number of the changes of a cname record for a seed domain. This feature was obtained in order to separately cluster domains that have stable cname records, as compared to domain with a more dynamic cname record which may be an indicator of malicious activity because by intuition malicious domains are more likely to switching their cname records at a high rate in order to avoid detection and subsequent blacklisting.
   c) *Average number of cname records per query* (Label: avg_n_cname_per_query)
   This feature gives the average number of cname records per query.
3) **Geographic features:**
   a) *Geographical location of the seed domain*
   We use MaxMinds python based GeoIP package, to obtain the geographical location - country code - of the seed domains. This feature is to provide a demographic based host server distribution of the domains with respect to their ranks. This when used in conjunction with the following feature provides us with a better geographical clustering of

seed to secondary domain pair based on the class of the seed domains, i.e. clustering of domains with a global presence to a local presence.

b) *Variability of the geographical location of the secondary domains*
This is a key feature that compares the geographic location of the seed domain with the geographic location of the secondary resolutions. From [2] it was observed that high ranked domains had a large demographic spread of their secondary resolutions, which is expected due to their global presence. Middle to lower ranked benign Alexa domains however had a strong correlation between the secondary domain resolutions and their seed domain, i.e the locations of the secondary domains are more likely to be within the same country as the seed domain. This can be attributed to these domains association with small businesses and localized groups, often with a domestic reach only. Any deviation from this high correlation of location, especially for the lower ranks, can be attributed to a botnet communicating with their Command & Control.

Using the above feature set we were unable to get a strong clustering of Alexa's domains. Hence it became imperative to utilize additional feature sets to produce a more meaningful and distinct domain clusters. These additional feature set are cited below and are self explanatory in nature.

- *Alexa Rank*
- *Number of unique queries* (Label: n_unique_queries)
- *Average length of the queried URL* (Label: avg_query_length)
- *Number of SLD also cited in Alexa 1M* (Label: n_slds_in_top_1m))
- *Alphanumeric and special character distribution in the SLDs*
- *SLD belongs to 'WWW subdomain, a CDN or an advertisement dissemination servers*
- *Appearance on Public Blacklist (either IP address or Domain)* (Label: b_hit_blacklisted_domain & b_hit_blacklisted_ips)

B. *Classification*

Based on our feature selection we initially tried to cluster/classify domains by their Alexa rank or identify their approximate rank based on their DNS behavior. The results were inconclusive as we observed that the DNS infrastructure was largely similar irrespective of the rank a domain might hold, making accurate classification an implausible approach.

This led us to change our systems immediate goals. We used the previously created behavior model and trained it against known blacklisted domains and IPs present in our training data. When the previously illustrated feature set, in conjunction to the blacklist data was, re-modeled and and the system trained, it was able to effectively classify other blacklist domains from the testing data based on their DNS behavior.

The process by which we developed our classifier was as follows: we compiled our data into a feature matrix (2D array of samples against their feature values) and normalized the data using Python's standard scaler. We then performed model selection by plotting each classifier's ROC curves and choosing the best one. In our particular case, decision tree-based AdaBoosting appeared to work best and performed about as well as Random Forest, which is intuitively understandable. Finally, we evaluated the effectiveness of the individual features with regards to their predictive power.

V. RESULTS

The models we built showed continued improvement with the creation and inclusion of additional feature characteristics, albeit slowly. We first attempted blind clustering of the data to observe any obvious patterns, performing several experiment iterations with an array of algorithms. With the feature vectors chosen, we found k-means clustering algorithms to provide the most useful information (in particular, the MiniBatch K-Means library provided by sklearn; traditional K-Means proved to be computationally prohibitive for our resources).

Unfortunately, this yielded no noteworthy results. Abandoning the unsupervised learning algorithms, we next performed several binary classification experiments on manually labeled data, hoping to score a domain's Alexa rank based on it's resolved DNS characteristics. Figure 5 shows limited success with this approach: perhaps not surprisingly, domains at the very top of the Alexa 1M list have much in common and are easier to identify. As the rank decreases, our classifier performance steadily declined. After two major failed experiments, we looked towards cross referencing public domain and IP blacklists to label questionable seed domains, which finally provided us with hopeful results.

We were able to successfully predict whether a domain was blacklisted or not based on the dns traffic. We tested many different classifiers and had good success with most of them, although the Random Forest Classifier and AdaBoost Classifier outperformed the others, as shown in our ROC curve below. These malicious domains typically requested and resolved less CDNs than the rest of the Alexa Top 1M, making them easier to classify.

TABLE I
KEY STATISTICS OF THE POSTGRESQL DATABASE

Statistics	10-19-15	10-28-15	Total
No. of seeds	152970	186313	339283
No. of SLDs	358593	422157	780750
No. of IPv4	608168	697932	1306100
No. of IPv6	52193	59146	111339
No. of cname	131014	146325	277339
No. of requests	1817481	2211467	4028948
Time to download	126.32 s	167.29 s	–
Time to extract features	117.56 s	130.31 s	–

TABLE II
RANDOM FORREST CLASSIFIER STATISTICS

TP Rate	85.89%
FP Rate	14.11%
Accuracy	83.14%
Precision	85.05%
Recall	80.39%

TABLE III
FEATURE IMPORTANCE

avg_n_ipv4_per_query	31.88%
avg_n_ipv6_per_query	8.17%
avg_query_length	30.28%
n_unique_queries	0.0%
n_slds_in_top_1m	6.12%
n_ipv4	7.42%
n_ipv6	0.67%
n_cname	2.15%
n_distinct_slds	3.75%
avg_n_cnames_per_query	9.05 %
n_dns_queries	0.52%

VI. FUTURE WORK

In the future, we hope to eliminate Docker from our system because of its flaws, possibly switch to a Neo4j database for scalability, add more features, and look into how traffic changes over time to detect anomalies. Docker had issues in terms of freeing up resources (for examples, when a child process has an error and exits, the associated docker container remains alive, thus holding on to those resources), left a large memory footprint, and simply was not a fully implemented program. If the features it claims to support are ever fully integrated then it may remain viable to the system.

We may switch the database to Neo4j after a full comparison of the two is complete. For this stage of the project PostgreSQL worked well once it was properly indexed, but Neo4j may allow our system to be more scalable. We want to look into adding more features and removing features that do not have much importance (rather the latter features have the negative effect of adding noise to the system). This will hopefully increase our accuracy and reduce the number of false positives. We would additionally like to investigate how DNS traffic changes over time and whether that can be used to determine new features and detect anomalies. This was infeasible at the current state of our project due to our inability to crawl in a more time efficient manner. We believe that we were mainly limited by our network bandwidth, and may be able to eliminate this bottleneck by using a server that has a large, dedicated bandwidth (such as Amazon Webserver).

In this research, we have not evaluated DNSg against possible evasion techniques. This may involve adversaries planting elements in their domain infrastructure in order to bias important features used by DNSg. Another possible method that an adversary may use, would be to avoid designing a malicious webpage that is similar in nature to webpages in public blacklists. The feasibility and effectiveness of using any such techniques for evading DNSg has been considered as future work.

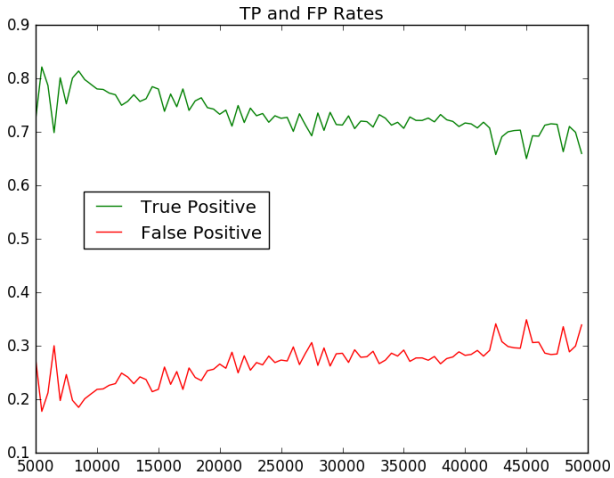

Fig. 5. The figure shows the probability of true positive and false positive for DNSg, for classifying higher Alexa ranks

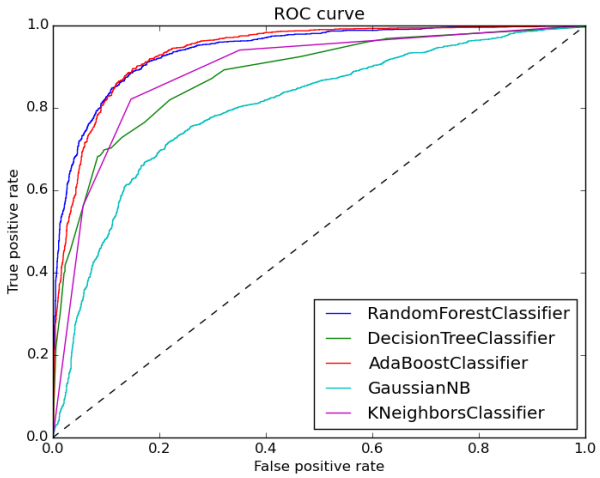

Fig. 6. Malicious Domain ROC

VII. CONCLUSION

A large corpus of information can be obtained by analyzing secondary DNS resolutions generated when querying a website. By leveraging the information we gained through scraping sites in the Alexa 1M, we were able to construct a robust classifier to detect blacklisted domains and IPs.

Our initial feature set and classification proved inadequate to seriously address our original objective, which was to accurately infer a websites Alexa rank by analyzing the DNS traffic it generates. It became apparent that structurally very similar sites can be widely dispersed by rank, while two similarly-ranked sites can exhibit very different traffic patterns. This naturally suggests a possible avenue of future work, to determine whether such a system can be made

effective.

However, we were able to augment our feature set and repurpose our model to effectively identify malicious (blacklisted) domains. This permits the possibility of deducing meaningful facts about domains on the basis of relatively superficial and quickly obtained information, which recommends our system as a supplement to more resource-intensive and robust analytical methods.

REFERENCES

[1] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a Dynamic Reputation System for DNS", USENIX Security Symposium, 2010.

[2] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou and D. Dagon, "Detecting Malware Domains at the Upper DNS Hierarchy", USENIX Security Symposium, 2011.

[3] S. Hao, N. Feamster, and R. Pandrangi, "An Internet Wide View into DNS Lookup Patterns", http://labs.verisigninc.com/projects/ malicious-domain-names.html, 2010.

[4] B. Zdrnja, N. Brownlee, and D. Wessels, "Passive monitoring of DNS anomalies", In Proceedings of DIMVA Conference, 2007.

[5] P. Mockapetris. Domain Names - Concepts and Facilities. http://www.ietf.org/rfc/rfc1034. txt, 1987.

[6] P. Mockapetris. Domain Names - Implementation and Specification. http://www.ietf.org/rfc/ rfc1035.txt, 1987.

[7] Christopher Olston and Marc Najork,"Web Crawling", Foundations and Trends in Information Retrieval Vol. 4, No. 3 (2010)

[8] M. Wu and J. Lai,"The Research and Implementation of parallel web crawler in cluster", International Conference on Computational and Information Sciences, 2010.

[9] M. Najork and A. Heydon, "On high performance web crawling", Compaq Systems Research Center, 130 Lytton Avenue, Palo Alto, California

[10] R. Chamberlain, J. Schommer, "Using Docker to Support Reproducible Research", figshare, 2014

[11] http://www.uxworld.com/?p=384

[12] Z. Qian, Z. Mao, Y. Xie and F. Yu, "On network- level clusters for spam detection", In Proceedings of the USENIX NDSS Symposium, 2010.

[13] R.Perdisci, W.Lee, and N.Feamster., "Behavioral clustering of http-based malware and signature generation using malicious network traces", In USENIX NSDI, 2010.

[14] S. Garera, N. Provos, M. Chew, and A. Rubin, "A framework for detection and measurement of phishing attacks", In Proceedings of the ACM WORM. ACM, 2007.

[15] S. Hao, N. Syed, N. Feamster, A. Gray and S. Krasser, "Detecting spammers with SNARE: Spatio- temporal network-level automatic reputation engine", In Proceedings of the USENIX Security Symposium, 2009.

[16] D. Dagon, C. Zou, and W. Lee, "Modeling botnet propagation using time zones", In Proceedings of the 13th Network and Distributed System Security Symposium NDSS, 2006.

[17] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Exposure: Finding malicious domains using passive dns analysis", In Proceedings of NDSS, 2011.

[18] D. Anderson, C. Fleizach, S. Savage, and G. Voelker, "Spamscatter: Characterizing internet scam hosting infrastructure", In Proceedings of the USENIX Security Symposium, 2007.

[19] S. Sinha, M. Bailey, and F. Jahanian, "Shades of grey: On the effectiveness of reputation-based blacklists", In 3rd International Conference on MALWARE, 2008.