BlackThread: Predicting malicious intentions

Rohit Belapurkar rbelapurkar3@gatech.edu
Ashwin Chintaluri achintaluri3@gatech.edu
Evan Downing edowning3@gatech.edu
Akshay Gupta agupta402@gatech.edu
Deepti Kochar deepti.kochar@gatech.edu

Haoran Ma hma@gatech.edu
Yash Shah shah_yash10@gatech.edu
Liyang Wan liyangwan@gatech.edu

***Abstract*—Classifying websites on how malicious or benign they are in nature is not an easy task. Many fall victim to visiting a malicious website that has the intent on downloading malware onto the victim's computer to perform illicit activities. Unfortunately, many antiviruses (AV) cannot prevent someone from visiting a malicious website. Web browsers even have a difficult time determining the intent of a website. Public blacklists for malicious domain names exist, but these seem to have fallen short of effective [9].**

We propose Blackthread, a comprehensive toolkit that allows its user to determine the intentions (i.e., benign or malicious) of any given website. This toolkit is comprised of many different analysis techniques in order to give a thorough examination of any given website. It also includes a tool to determine how closely related a website's content is to other categorized websites, such as sports, news, educational, etc.

Our results show that our toolkit is effective, with ROC curves with areas of 93% for analyzing URLs and 96% for analyzing the structure of websites among other high values of success.

I. Introduction

Determining if a website is benign or malicious is an important problem that has been studied by many researchers for quite some time. Some have relied on analyzing the contents of the websites [7], [15], while others analyzed the JavaScript contained in the website [4], [8], the types of HTML responses and HTML code of websites [13], the properties of iframes [10], and even how malicious landing sites redirect a user to a specific page [14].

Noticing that, in general, positive results have stemmed from their works, we proposed combining the many different types of website analysis techniques into one easy-to-use toolkit to more accurately label a website as having benign or malicious intent. We argue that this toolkit should be easily modified to accommodate future analysis methods. This toolkit, Blackthread, is our proposed solution to the problems described above.

Blackthread uses multiple machine learning techniques to classify each type of analysis in order to increase accuracy and confidence in the labeling of a website as either benign or malicious. Taking each analysis technique's classification of a website separately, one can comprehensively reason about the intentions of a website.

As an added feature, Blackthread also includes the ability to score website contents based on a particular themed category (i.e., Sports, Energy Industry, Defense Industry, etc.). This feature has nothing to do with classifying the benign or malicious intentions of a website, but is rather an additional feature that can be used to reason more about the particular contents included on a website.

Next we will describe in detail how Blackthread is organized and how the rest of this paper will follow.

II. Methodology

The primary aim of the Blackthread toolkit is to provide various utilities that can predict whether a given site is malicious or benign in nature. Each of the utilities provided in the toolkit classify, in parallel, whether it believes a website is benign or malicious based on the classification models used for each of the given utilities. We will collectively call these utilities as decision utilities from this point forward. The toolkit also provides various base utilities that are primitive to the operation of the decision utilities. Also included is a ranking utility used to produce a score on how related a website is to a certain topic (i.e., Defense Industry, Energy Companies, Sports, etc.).

We give brief descriptions of the base utilities at the end of this section and detailed descriptions of the decision utilities in a later section. For decision utilities we define the purpose of the utility, the details of the databases used to generate the knowledge base required for each utility to make future predictions, and the operations that were performed on the knowledge base to construct a model for the malicious activity targeted by that specific utility.

Each of the decision utilities in the toolkit focus on a specific type of malicious activity that is prevalent on the Internet. Most of these activities are extensively studied in literature. We will label the utilities implemented in this toolkit based on the aspect of the Internet they analyze. Hence, the decision utilities in the Blackthread toolkit are: (1) Web structure agility, (2) Lexical properties of URLs, (3) iFrames and JavaScript, and (4) DNS agility.

These four decision utilities form the classification prowess of the Blackthread toolkit. Each of these utilities is autonomous in nature and can independently classify a given URL based on the individual knowledge base. These four utilities are supported by the different base utilities that provide different functionalities that aid in constructing the classification models for the decision utilities. Some of the primary functionalities provided by the base utilities include data collection through crawling websites, cross-validation, and model selection. The base utilities in the Blackthread toolkit are as follows: (1) Crawling and (2) Classifier Generation. We will discuss these base utilities shortly.

The rest of this paper is organized as follows: Section III describes the decision utilities in detail, Section IV describes the ranking utility in full, Section V discusses our results, and Sections VI and VII describe the related works and conclusion respectively.

We now briefly describe each of these base utilities before describing the decision utilities that use them.

A. *Base Utilities*

Crawling websites is a necessary capability of our toolkit in order to gather, also known as "scrape", data off of websites in an automated fashion. The data gathered from the contents of a website can contain anything from the plain text to the tags and scripts that give the website its structure. We used Scrapy, a python-based webcrawler, to crawl and store data contained on websites quickly. We created four different spiders for crawling contents of websites:

- Tag Spider: Collected HTML tags on the website
- Content Spider: Collected the textual content on the website
- URL Spider: Collected the URLs on the website
- Script Spider: Collected the iFrame tags and JavaScript code on the website.

Our spiders use rotating user agents, download delays, and a cookie disabler to prevent our spiders from getting blacklisted. During this project, not once did we get blacklisted by any website, even after extensive crawls. We use a pipeline function used by the Scrapy interface to record collected data to databases quickly. We defined 3 different types of item categories to record specific data on websites:

- Tag Item: Includes the tag, the URL it came from, and the position it was in on the HTML page
- Frame Tags: Includes the iFrame ratio, the iFrame itself, the JavaScript ratio, the link ratio, the contents of the JavaScript, and the URL the data came from
- Script Tags: Includes the JavaScript and the URL the script came from

We used a number of different classification algorithms to label our extracted features. Specifically we compared the effectiveness of a decision tree, support vector machine, logistical regression, random forest, gradient boosting, and gradient decent algorithms. These algorithms are contained in Sklearn [1], a python library containing implementations of popular machine learning algorithms. Using 10-fold cross-validation, model selection, Receiver Operating Characteristic (ROC) curves, precision, recall, and accuracy attributes of the learning algorithms, we were able to classify benign and malicious websites effectively.

When viewing our ROC graphs, note that [5] says that if the curve is located in the top left corner of the graph and the precision curve is located in the top right corner of the graph, then the classifier used was good.

III. Classifying Websites

A. *Web structure agility*

When studying the properties of malicious websites, a common property of fast-flux is persistence in most of these websites. Fast-flux websites regularly change some of the properties of their pages or infrastructure to avoid being detected or taken down by law enforcement. This property can also be called the agility of the site. Typically benign websites with a legitimate business model do not change the infrastructure or other fundamental properties of their site as frequently as malicious websites. Legitimate websites do this in order to ensure stability of the site and reliability of the business. A benign site can, however, frequently update the contents of the pages on its website. This behavior is most common in news bulletins that frequently update their pages with new information.

We felt a need to analyze this agility of a malicious website in order to classify future sites with a similar sort of agility. We decided to analyze the structure of a web page for agility; that is, we measure changes in the HTML tags of a web page and their position over time. A benign site will update its contents regularly but it will not update the container of those contents. On the other hand, a malicious site can discard the entire HTML structure of the page. We will now describe the data collection process done for constructing the knowledge base of the classification model that is used to predict whether a site is malicious based on its structural agility.

1) Data Collection: We selected a list of 50 known malicious websites from a website that collects malware domains to server as a blacklist [2] and a list of 25 benign sites from Alexa where each URL had a popularity rank between 426-500. These 75 URLs were used to construct the feature vectors for our machine learning classifier. The reader at this point will notice a bias in our ground truth towards malicious websites. This bias was not intended but was rather introduced due to the challenges faced in data collection, which we will describe in Section III-A3.

We crawled these 75 URLs in regular intervals of 12 hours for 15 days. HTML tags were extracted sequentially from the crawled pages and each crawled website was stored in separate databases for each time interval.

[1]scikit-learn.org/stable

[2]www.malwaredomainlist.com

During the crawling process we noticed that the size of the website did not affect the size of the database of tags extracted from the website. Most of the tag databases were under 10 megabytes in size even when the size of the website was, suggesting that those websites contained more content than tags. The small size of the tag databases played a key role in feature extraction for training the classifier.

2) Feature Extraction: One of the most cumbersome and crucial steps in any learning algorithm is extracting features from a given knowledge base. The learning algorithms are mathematical in nature and thus require vectors to be numerical and not in textual format. This posed quite a dilemma for our implementation because all the data crawled from the URL's in the knowledge base was in text format and we required an efficient mechanism to convert this text data into numerical vectors in such a manner that the agility of the tags of the web pages is captured by the numerical vector.

We scourged the literature to find a mechanism that could achieve the goal defined above, but we could not find any function that was tailored for this problem. In the end we looked at string matching algorithms and found an effective statistic the could capture the change in tags between two given web pages. We used the function called the Levenshtein distance also known as the edit distance. Edit distance is the number of characters that need to be changed, added and deleted from one string to make it equal to another string we are comparing.

Thus we decided to form our feature vector in such a manner, we competed the edit distance between the tags of a websites that were crawled after 12 hours, we repeated this process 7 times and covered 8 such crawling instances to create a 7 element feature vector for each of the URL in the knowledge base.

We then provided this ground truth of feature vectors to the "Classifier Generation" base utility implemented in the Blackthread toolbox, this utility generated a statistical report of false positive and true positive rates under the given ground truth database for different popular classifiers, the functioning of this base utility is described shortly, while the report generated by this utility and the classifier selected for online prediction is described in the next section.

3) Challenges: While the technique described above it might seem trivial, it came with its own sets of challenges that were non trivial to deal with, however these challenges do not provide any significant contribution to the readers understanding of our technique, they reflect the journey taken during the implementation of Blackthread.

One of the very first major challenge faced by us was during crawling of the URL's in the knowledge base, crawling is an automated process and practically should be much faster than a human clicking through the web pages. In our implementation the crawling process initially was very sluggish, this was due to the fact that we were writing the data crawled directly into the database without maintaining any queues. We introduced queues when writing into databases to decouple the crawling and database engine, this boosted the crawling process to unprecedented speeds and the URL's that took us hours to crawl initially now took less than 10 minutes to crawl.

Another major challenge we faced was when we started calculating the edit distance of two tag databases, the best complexity achieved by the edit distance function in literature is order of multiplication of length of the two strings, in this case the length of the strings was typically in kilobytes or megabytes of length, thus the running time and memory requirements of the algorithm are very high given the size of the database.

We implemented several different optimization techniques in order to preprocess the databases before feeding them to the edit distance function, some of the optimizations include using hash functions to remove similar pages in the two database, adding the length of new or deleted pages directly without calculating the edit distance, even with all these optimizations it takes approximately 8 hours to calculate the edit distance between two instances of the benign URL dataset from the knowledge base. This slow process of calculating the edit distance was also part of the reason why so few malicious and benign websites were crawled (as there would not have been time to classify tens-of-thousands of websites), not to mention many of the initial malicious websites in our study were taken down after a few days. Even then, crawling and recording the complete contents of websites takes time and space which further extends the time of calculating the edit distanc. It was also difficult to gather the contents of still-existing malicious websites. However, as our results show, even for the small amount of websites crawled it seems that our classifier had enough information to effectively classify the differences between a benign website's structure and a malicious website's structure.

We earlier mentioned the bias in our knowledge base, this bias was introduced because the edit distance function occupies too much memory even for databases of sizes in few kilobytes, we noticed that if the size of the database crawled was greater than 10MB then the Linux machine will automatically kill the edit distance function at some point of time, this size limit restricted us to a very small benign URL dataset from the Alexa top 500 websites, on the other hand many malicious URL's easily came under the size limit, thus causing a bias in the knowledge base.

B. Lexical properties of URL

Adversaries use the Internet to carry out their malicious activities. These activities include spamming, phishing, spreading malware through drive by downloads, etc. To avoid detection, adversaries constantly change their domain names which effectively makes blacklisting useless. Blacklisting domain names, although it results in no false positives, is inefficient to tackle this problem as by the time a domain name is recorded in a blacklist, it would have already carried out its malicious purpose. This weakness of blacklists can be overcome with the use of anomaly based detection methods designed to detect unknown malicious URLs. In this method, we classify the URLs as either being malicious or benign based on the distinct characteristic features of each of their class. In our work, we focus mainly on one type malicious activity—phishing.

Phishing is a technique where malicious users (termed as "Phishers") use a combination of tricks using e-mail, webpages and malicious softwares to steal a person's identity and financial credentials. Phishing typically involves sending an email from a seemingly trustworthy source to an unsuspecting user to click an URL contained in the email which links to a counterfeit webpage. This webpage usually resembles that of a well known brand or a financial institution (e.g., Wells Fargo Bank) and contains fields for private information about the user's financial data such as credit card number of their bank account number and password. An unsuspecting user may fall prey to these techniques and end up giving their private information to Phishers who further use this data for carrying out malicious activity. Phishers may themself use this data to carry out unauthorized financial transactions masquerading to be the authentic user or they may sell this information in the underground market.

In our work, we propose to exploit the unique characteristics of phishing URLs to classify them as either being malicious or benign. Phishing URLs and domain names have some characteristics which would differ from those of normal URLs and hence could be used to classify them. Phishing URLs and domain names have different lengths as compared to normal benign URLs [3].Even their character frequency is different than those of benign URLs. Also, a common feature of phishing URLs is that they usually contain the name of the brand they are targeting (e.g., Wells Fargo) in their URL [11].

1) Data Collection: For our work, we collected both phishing URLs as well as benign URLs. Our primary source of phishing URLs is PhishTank [3] which contains a database of phishing URLs either submitted by users (and further verified) or obtained from external feeds. This database is constantly updated every hour. For our work, we considered a database of approximately 10,000 phishing URLs obtained from PhishTank with about 60% of them being targeted on financial institutions and well known brands (e.g., Apple). For our database of benign URLs, we used Alexa's top websites crawled by our crawler. We only considered those Alexa websites that had ranking high scores. Our intuition was that a website will not be popular unless and until it is completely legitimate and not malicious.

2) Feature Extraction: Our work on URL analysis is based on the fact that phishing websites have a domain name and URL structure that is characteristically different from those of legitimate and benign URLs. These characteristics of phishing URLs can be exploited to differentiate them from the benign URLs. For the purpose of classification, we developed a feature set consisting of 12 features which could be characteristically unique to the phishing URLs. Our feature set includes presence of brand names in the path, URL length, domain length, domain token count, average domain token length, maximum domain token length, path length, path token count, average path token length, maximum path token length, character frequency, frequencies of special characters in the path.

[3]www.phishtank.com

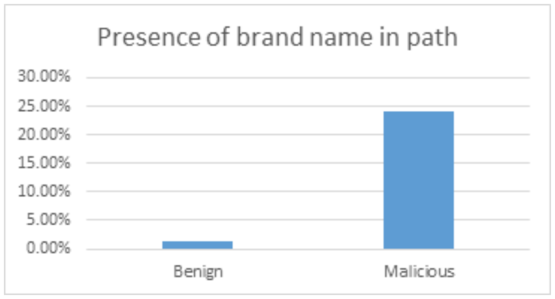


Fig. 1: Percentage of URLs having brand name in their path.

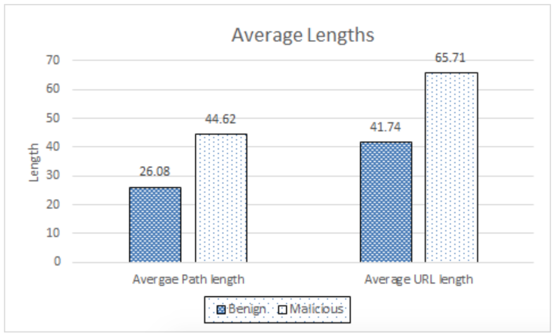


Fig. 2: Average path and URL lengths for benign and malicious URLs.

We chose to search for the brand name in the URL because, as illustrated in Figure 1, the probability that a well known brand is present in the path of a malicious URL is higher than the probability that a benign URL contains the brand name in its path. Generally, the brand name is present in the FQDN of the benign URL.

We give the same defense for using the average length of the URL in Figure 2, as we can see that the average URL length and average path lengths of a malicious and benign URL differ from a substantial margin. These features can be used effectively for differentiating a malicious URL from a benign URL.

Every URL consists of two main components: the fully qualified domain name (FQDN) and their path. As mentioned earlier, phishing URLs often contain the name of the brand they are targeting. However, this brand name is usually contained in the path rather than in the FQDN as is the case with the benign URLs. Another feature is the frequency of vowels and special characters in the path. The paths, in case of benign URLs, usually resemble the English language and makes use of dictionary words. This gives them a specific character frequency which is also shared by phrases in the English language. This is not the case with phishing (malicious in general) URLs as they often have a path which makes use of random letters and words and hence do not have the same character frequency as the benign URLs. Further, the paths in phishing URLs have a higher frequency of special characters and numbers than the paths in benign URLs. For example, let us consider an actual phishing URL from our phishing URL database [4]. This phishing URL was active when we analyzed our work but was later taken down. This URL specifically targets the Wells Fargo Bank [5]. As we can see, the name of the brand is contained in the path rather than in the FQDN as in case of the authentic institution. Also, the phishing URL consists of a path which has a higher frequency of numbers and random letters. If the frequency of the vowels is compared to the frequency of the other letters (e.g., 'f', 'b', etc.), we find that they are comparable which is not the case with benign URLs which generally have English words in their paths.

Using the feature set described above, we carried out tests on all the phishing URLs present in our database and collected data pertaining to every feature in the feature set and the results were stored in database. The same set of tests were carried out on benign URLs leading to another set of results which were stored in a separate database. Our goal is to classify the phishing URLs from the benign URLs and the most effective way was to use a machine learning classifier. The classifier was trained on a set of results obtained from analyzing the phishing URLs and also on the set of results obtained from our analysis of benign URLs. We summarize the results of the classifier and the database in the analysis section of this report.

3) Challenges: The main challenge we faced was that of URL shortening services such as tinyurl, bit.ly, etc. Such URLs tend to have a structure which is different from both the benign as well as the malicious URLs. A possible way around this is to learn the mapping of the shortened URL to its original domain name and carry out analysis on the domain name. This could be a possible case of future work in our project. For the present, we have not considered the URL shortening services and their possible analysis in our work.

Our work revolves around effective identification of malicious URLs which do not make their way into blacklists or do so when they have carried out their malicious activities. However, our analysis of URLs is biased towards phishing URLs. Phishing URLs, although malicious, have an extra characteristic feature which the malicious URLs generally do not have. Phishing URLs contain the name of the brand they are targeting in their URL path and this changes the character and letter frequency statistics and this feature makes them characteristically different from the rest of malicious URL types.

C. iFrames and JavaScript

Web content analysis isn't usually an effective indication of the malicious or benign intent of a domain. However, analyzing the iFrames and JavaScript (JS) of a website is a great measuring stick of the true intentions of a website. Popularly ranked and legitimate websites generally have a tight control over the content they allow on their sites. Benign websites avoid misguiding the user through advertisements (ads) or other malware, whereas malicious sites try to impose upon the user some sort of malware or ads through the use of iFrames and some malicious JavaScript (JS) calls. iFrames can be used for both benign and malign purposes, but when used for malign intent, a certain anomalous behavior in their attributes is evident. Our aim is to extract the attributes of the iFrame and the embedded Javascript functions on webpages to generate a score based on the content. This score is then compared to a threshold pre-calculated from analyzing various benign websites. Then the website in question is labeled as malicious or benign.

1) Data Collection: For our data , we extracted the iFrame and JS features by crawling a known list of about 15,000 malicious websites and a list of benign websites from Alexa and created a database for our analysis.

2) Feature Extraction: The following features were extracted and analyzed to generate the score:

- iFrame Tag Ration (iTR): iTR is the ratio between number of iFrame tags and total number of tags. It indicates the percentage of volume or weight iFrame in a given webpage. If iTR value h is significant higher than the threshold k, we suspect the page might be malicious. The threshold will be set specifically during our experiment.
- iFrame Zero-Size Counter: Injecting hidden malicious iFrames into compromised legitimate websites is one of the most popular types of malicious attacks. When the user clicks on the page, the actions in the iFrame are triggered along with the intended action on the actual webpage, thus causing a security problem. An iFrame is a rectangular element of webpages where you can load other web pages either from the same site or from third-party sites. Tricks which attackers often use to hide iFrame are modifying the element's zero-length sides, visibility, and JavaScript onload. We count the number of zero-size iFrames in a given web page and store it as Zero-Size Counter.
- iFrame JS Ratio (iJSR): The main purpose of an iFrame is to display other web page either from internal or external sources. The most common use of iFrame is to display ad blocks. If iFrame contains only JavaScript, it is seen as quite suspicious. JavaScript in an iFrame might contain function calls or attributes that are anomalous handles which might cause unintended triggers. Therefore we evaluate the ratio of length of JavaScript to the length of iFrame itself. If JiR is high, there might be a malicious intention behind the iFrame.
- iFrame JS function calls: Often, the Javascript in an iFrame might contain function calls or attributes that are anomalous handles which might cause unintended

[4]Please do NOT click this: blacktiemartiniclub.com/wp-content/Wellsfargo/d39069817603deb9bc34f10b2520189e

[5]www.wellsfargo.com

triggers. There is a need to parse the JavaScript in the iFrame to check for malicious triggers, e.g., autorun.exe , that automatically triggers the unknown script. The function calls used in our analysis were escape(), unescape(), eval(), setTimeout(), exec(), search(), and unbound(). Also, if a JavaScript performs encapsulation, decapsulation, redirect, and post-back in then the page might be malicious but due to volume concerns these attributes were not taken into consideration in our analysis.

A total score is calculated by assigning weights to individual features and then taking the cumulative score. The weights are assigned based on the analysis of benign websites. Since the existence of zero-size iFrames is a high indication of malignity, it has the highest weight followed by malicious JavaScript function call count. The rest of the attributes are given lower weights. This total score is calculated for a set of known benign websites and a threshold is arrived upon based on the mean, median, and mode of the scores. This analysis is then run upon the known malicious list to determine their scores and successfully flagging them as malicious. The code was also tested on a mixed collection of benign and malicious websites and this resulted in good accuracy .

3) Challenges: The main challenge faced was to determine the weights to be assigned to individual features and the threshold for comparison. It is essential to have the correct priority of assigning the highest weights for decision of malign behavior as some of the chosen features like zero sized iFrame, etc. are not typically exhibited by benign websites. We iteratively selected the weights and ran on a known set of benign and malicious sites to check the validity of the score generated. This helped us optimize the weights to be able to give a good distinction between the benign and the malign sites.

D. DNS agility

The Domain Name System (DNS) plays an important role in the network operation that forms the Internet, by providing a two-way mapping between the domain name and its associated IP address. Given this fundamental role, it forms an integral aspect of the Internet's architecture. This makes it a beneficial front to monitor domain related activities [2]. However, performing a passive agility analysis (measured changes over a specific metric) on this front poses its own set of difficulties as most domains resolve to a fixed set of IP addresses; exceptions being Content Delivery Networks (CDN) and fast flux domains usually associated with malicious domains. However, it is possible to derive some baseline characteristics based on the DNS information obtained, when it is queried, to resolve a domain to its associated IP address.

The information obtained can be used to classify and assign a domain's DNS related activity to be suspicious or non-suspicious. To derive these baseline characteristics and classifications we measure the agility observed in a domains DNS resolutions as viewed from the perspective of systems located in different countries. The changes measured are in two fronts, namely resolved domain IP and its associated Time-to-Live (TTL). By measuring these changes and classifying domains, we will be able to assign a specific the domain to a classification and based on the similarities or differences observed between the said domain and others within the same class we can attribute its DNS behavior to malicious or benign.

1) Data Collection: To create a working framework we selected ten domain names belonging to the following classes:

- Famous Domains (e.g., Google, Facebook)
- Common Domains (local or regional domains with little global presence)
- Content Delivery Network (e.g. Akamai, AWS)
- Malicious domains.

We included malicious domains in this mix so that we can visualize the similarities and distinctions between the three benign domain classes and known malicious domains. This list was created using the Alexa top 500 list, a malware domain list [6], and the Arbor Atlas [7] fast-flux domain list. To resolve the domains using recursive DNS (RDNS) we randomly selected 100 servers located in different countries using the Public DNS Server List [8]. We build a database using the first two lists and queried each DNS server in the third list requesting resolution of a domain name. This can be accomplished using a DNS resolver python-based library, but a Unix system command dig can also be used to achieve the same results. We queried this set of Domain-DNS mapping at intervals of 12 hours over three days to obtain a set of Domain-IP resolutions from one hundred different countries.

2) Feature Extraction: The following features are utilized to establish the nature of a given domain. These features are inspired from [12].

- Number of resolved IP's: This is the overall number of distinct IP addresses obtained for a given domain, each time it is queried against our set of DNS servers.
- Average time-to-live: The average Time-to-Live observed for each distinct IP for a given domain

3) Challenges: A major challenge faced during the analysis was due the absence of an active malicious fast-flux domains needed to conduct a comparative study between nature of a malicious and benign domain. This challenge was mainly present due to the short active life of malicious fast-flux domains. We were unable to obtain references to active malicious fast-flux domains and any reference we obtained was pointing to an already inactive domain name [9] , [10].

[6]www.malwaredomainlist.com/mdl.php
[7]atlas.arbor.net/summary/fastflux
[8]public-dns.tk
[9]www.honeynet.org/node/138
[10]mirror1.malwaredomains.com/files/domains.txt

IV. Ranking Utility

Our ranking utility is the final component of our Blackthread toolkit. It has the ability to analyze the textual contents of a website and produce a score that shows how similar it is to categorized websites. That is, it can determine the difference between a website that services the energy industry and one that services the defense industry. This feature of our toolkit is meant to further distinguish characteristics about a given website that may not be apparent to the viewer. It is not meant to classify differences between benign and malicious websites, but is meant to classify differences between topical websites.

A. *Keyword Extraction*

In order to calculate a score representing how similar a website is to websites in certain categories, we gathered the contents of 7 defense industry websites and 7 sports websites and stored them into a database. Then, using a database parser and a python library containing the English dictionary, we counted the frequency of words used in the contents of the webpages and stored these results in a text file to be parsed by our rank calculator.

B. *Ranking Score*

After gathering the count frequencies of words that exist in webpages belonging to certain industry categories, we crawled the contents of a sports webpage to see how similar it was to a defense industry webpage. The normalized score showed that the two website were significantly different categorically.

C. *Challenges*

There were no challenges with building our ranking system. However, one problem that exists as a result of website ranking systems is "water holing." This type of attack focuses on the observing popular websites accessed by a certain organization. The attack then focuses on attack those popular websites in order to gain unauthorized access to the organization because of trusts that may exist between the organization and the popular websites it is involved with.

V. Results

In this project we considered different techniques for creating the classification model, from thresholding to random forests. We decided to use different models of classification for different decision utilities. In this section we describe the evaluation done for model selection for each decision utility and the give some statistics and ROC curves to support our classification model.

A. *Web Structure Results*

In analyzing web page structure, we used a ground truth containing 75 distinct URL's, 50 of which were from malicious sources while the other 25 were crawled from benign sources. Each of the feature vector in the ground truth consisted of 7 features which represent the edit distance between two consecutive crawls of a given web page, we crawled each website after 12 hour interval collecting crawling data for 4 days generating 8 data points for each website, from these data points we computed the 7 features for a feature vector to create the ground truth.

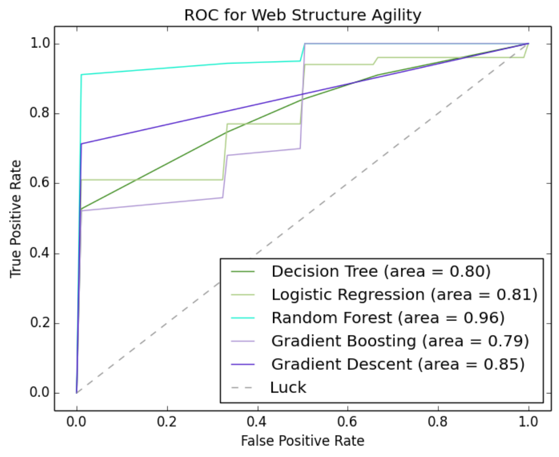


Fig. 3: ROC for web structure agility classification

On the ground truth we executed the classifier generation base utilities which automatically generated the ROC curves and the other evaluation statistics for six different classification algorithms. The utility did a 10-fold cross-validation to avoid skewing results and then generated the statistics and the plots to help us in model selection. The classification models that we considered are: (1) Support Vector Machine, (2) Decision Tree, (3) Stochastic Gradient Descent, (4) Gradient Tree Boosting, (5) Random Forest, and (6) Logistic Regression. The mean accuracy, precision, recall and area of the ROC curve for every model is given in Table I:

From Table I we can see that the Random Forest Classifier has the highest ROC area. That is, it reaches a true positive rate fastest as compared to other classifiers. We compute the other statistics to support the decision made using the ROC area statistics. In the above table we can see that Random forest classifier has an average precision and recall mean value and its not the worst or the best classifier based on precision and recall, but it is the best classifier based on ROC area, thus the classification model we chose to predict future URL is Random Forest classifier, we further support our conclusion by plotting the ROC curve f or the above classifiers as shown below (we could not plot the SVM classifier due to dubious memory errors):

From Figure 3 we see that the ROC curve for the Random forest classifier has the maximum area and grows much more quickly than the other classifiers, hence we chose the random forest classifier for classification in the web page agility decision utility.

	Accuracy	Precision	Recall	ROC Area
Support Vector Machine	0.79 (+/- 0.16)	**1.00 (+/- 0.00)**	0.69 (+/- 0.22)	0.84 (+/- 0.11)
Logistic Regression	0.79 (+/- 0.16)	0.96 (+/- 0.09)	0.73 (+/- 0.22)	0.81 (+/- 0.19)
Random Forest	0.82 (+/- 0.14)	0.90 (+/- 0.11)	0.85 (+/- 0.16)	**0.92 (+/- 0.11)**
Decision Tree	0.80 (+/- 0.14)	0.88 (+/- 0.12)	0.89 (+/- 0.17)	0.82 (+/- 0.15)
Stochastic Gradient Descent	0.80 (+/- 0.16)	1.00 (+/- 0.00)	0.71 (+/- 0.22)	0.87 (+/- 0.16)
Gradient Tree Boosting	**0.83 (+/- 0.08)**	0.84 (+/- 0.09)	**0.94 (+/- 0.09)**	0.80 (+/- 0.20)

TABLE I: Web Structural Agility - Model Selection

B. *Lexical Results*

In our lexical analysis we used a ground truth containing 18,000 distinct URL's, 9,000 of which were from malicious sources while the other 9,000 were crawled from benign sources. Each of the feature vector in the ground truth consisted of 26 features which are described in the Section III-B2.

On the ground truth, we executed the classifier generation base utilities which automatically generated the ROC curves and the other evaluation statistics for six different classification algorithms. The utility performed a 10-fold cross-validation to avoid skewing results and then generated the statistics and the plots to help us in model selection. The classification models that we considered are: (1) Support Vector Machine, (2) Decision Tree, (3) Stochastic Gradient Descent, (4) Gradient Tree Boosting, (5) Random Forest, and (6) Logistic Regression. The mean accuracy, precision, recall and area of the ROC curve for every model is given in Table II:

From Table II we can see that the Random Forest Classifier has the highest ROC area that is it reaches a true positive rate fastest as compared to other classifiers. We compute the other statistics to support the decision made using the ROC area statistics. In the above table we can see that Random forest classifier has the best precision and average recall mean value and its not the worst or the best classifier based on recall, but it is the best classifier based on ROC area, thus the classification model we chose to predict future URL is Random Forest classifier, we further support our conclusion by plotting the ROC curve for the above classifiers as shown below (we could not plot the SVM classifier due to dubious memory errors):

From Figure 4 we see that the ROC curve for the Random forest classifier has the maximum area and grows much more quickly than the other classifiers, hence we chose the random forest classifier for classification in the web page agility decision utility.

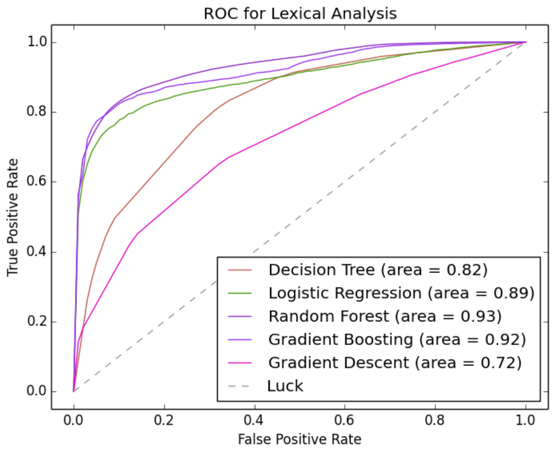


Fig. 4: ROC for lexical classification of URLs

C. *iFrame and JavaScript Results*

Each attribute is scored such that it falls between 0 and 1 and based on the iterations over many samples of data, the Weights were calculated as shows in the table below. The cumulative score is taken as the summation of individual scores multiplied by their respective weight fractions and therefore falls between 0 and 1 .This is then compared with the threshold and flagged malicious or benign . One main roadblock here was that there was a certain degree of overlap between the attributes of benign websites and those of the malicious websites. This was because, although a very high fraction of the benign websites exhibited low values for attributes such as Frame Ratio , there were a small portion of websites that are classified benign but have higher iFrame Tag ratios . Thus there was a need to categorize the websites are highly probably benign and highly probable malicious websites for certain cases. The maximum total score for Benign URL set is 0.33. It indicates that any URL with total score below 0.33 can be defined as benign. There are a certain high possibility that a URL with total score which is above 0.33. The minimum total score for Malicious URL set is 0.65. It specifies that a given URL with total score above 0.64 can be defined as malicious. Any URL with total score lands between 0.33 and 0.65 will be labeled uncertain.

Also , Redirection analysis of iFrames was not considered a feasible option as mentioned earlier , so this lead to having zeros on the JS ratio (JavaScript ratio) of the website as seen in Table III. Therefore the weight assigned for the JS Ratio is taken to be zero for now . This can serve as a placeholder for any future analysis when redirection analysis needs to be performed .

For the final analysis, a set of 2000 websites were crawled out of which 1000 websites were known benign and 1000 were known malicious websites. The known benign websites were taken from reliable sources such as Alexa and the known malicious websites were taken from sources like phish tank.

	Accuracy	Precision	Recall	ROC Area
Support Vector Machine	0.79 (+/- 0.16)	**1.00 (+/- 0.00)**	0.69 (+/- 0.22)	0.84 (+/- 0.11)
Logistic Regression	0.83 (+/- 0.10)	0.85 (+/- 0.14)	0.85 (+/- 0.03)	0.89 (+/- 0.09)
Random Forest	**0.86 (+/- 0.11)**	0.85 (+/- 0.14)	**0.94 (+/- 0.02)**	**0.94 (+/- 0.09)**
Decision Tree	0.82 (+/- 0.10)	0.80 (+/- 0.13)	0.92 (+/- 0.03)	0.82 (+/- 0.10)
Stochastic Gradient Descent	0.76 (+/- 0.14)	0.76 (+/- 0.12)	0.72 (+/- 0.16)	0.85 (+/- 0.11)
Gradient Tree Boosting	0.84 (+/- 0.10)	0.83 (+/- 0.14)	0.91 (+/- 0.02)	0.92 (+/- 0.11)

TABLE II: Lexical Analysis - Model Selection

	FrameRatio*10000	JS Ratio	Zero iFrame Count	Suspicious JS Function Call counts
Benign	0.70	0	0.73	3.71
Malicious	4.14	0	2.90	5.14
Threshold	1.43	0	1.82	4.43
Weights	25%	0%	40%	35%

TABLE III: iFrame/JavaScript Results

This list of websites were taken such that there is no overlap between these and the ones previously used for threshold calculations. The features for iFrame Tag ratio, JS ratio , Zero Iframe count and suspicious JS call count as mentioned above were crawled and the resulting attributes were stored in a database.

This database was analyzed and an individual attribute score and a total cumulative weights based score was generated and this was compared with the threshold value and the minimum and maximum thresholds of the malicious and benign thresholds respectively. Our software outputs if a given URL is malicious or not. Therefore we only consider the True Positive (The URL is a known Malicious and we clearly identified it) and False Positive (The URL is a known Benign and we improperly identified it as malicious) value of the experiment. We saw that the results gave a True Positive of 93% and a False Positive of 7% . Thus giving a true positive rate of 0.93. The False positive rate of this magnitude occurred , we believe, because of the unavoidable overlap area between the benign and malicious website attribute thresholds as discussed above.

D. DNS Results

The raw data we gathered by querying each Domain-DNS pair from our tables was used as a base data set to conduct comparative analysis between the set of benign and malicious domain names. Using this data set we extracted the following three features with the following results.

1) Number of Resolved IP Addresses: For all the resolved IP address obtained by querying out Domain-DNS name IP pair we extracted the number of distinct IP address obtained for the given domain name and the increment in the number of distinct IP's obtained after each query when compared to previous query's. We have separated the obtained data into two segments to differentiate the differences between domains with fast-flux properties and non-fast-flux domains. The data obtained and the changes can be visualized intuitively.

As observed from Figure 5, most of the observed IP resolutions are static with no observable changes beyond the first two queries (which is why we did not graph any queries after the second query). However, we can clearly see that the first three listed domains have higher degree when compared to other domains, of which "martuz.cn" is a known malicious domain.

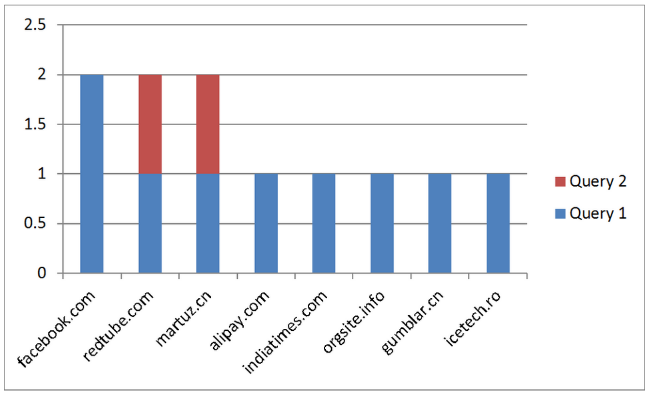


Fig. 5: Number of Distinct Resolved IP's for non-fast flux domains.

Though the above graph does not display any significant malicious activity, when analyzed alongside the data obtained from the average TTL analysis data, we observe abnormal behavior needing further analysis, to be discussed in a later section.

The three flux domains we observe in Figure 6 allow us to clearly visualize the difference between a fast-flux and a non fast-flux domain.

As we can clearly see, when compared to the previous figure, the number of IP's obtained is clearly magnitudes greater than non-fast flux domains, thus the need to analyze it separately. Here the first two websites are part of CDN's (Akamai and AWS) and the "google.com" domain maintains a global presence by hosting local servers to enable quick access, thus displays a fast-flux behavior.

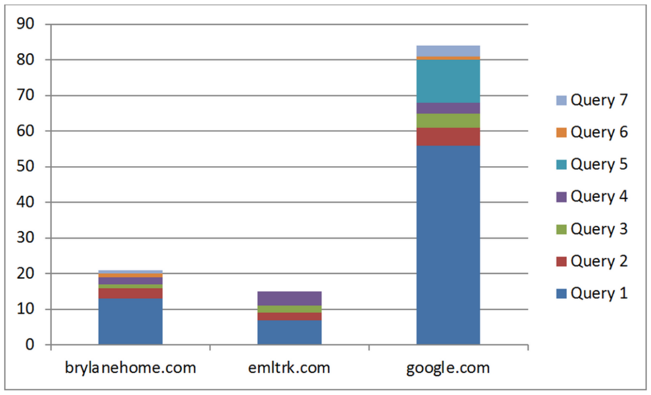


Fig. 6: Number of Distinct Resolved IP's for fast-flux domains.

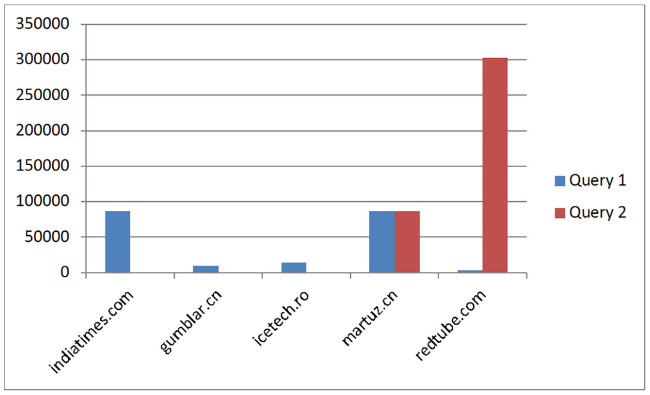


Fig. 7: Large TTL value domains.

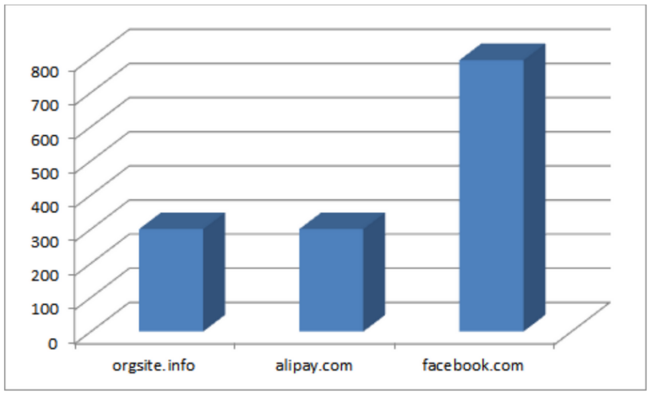


Fig. 8: Small TTL for domains.

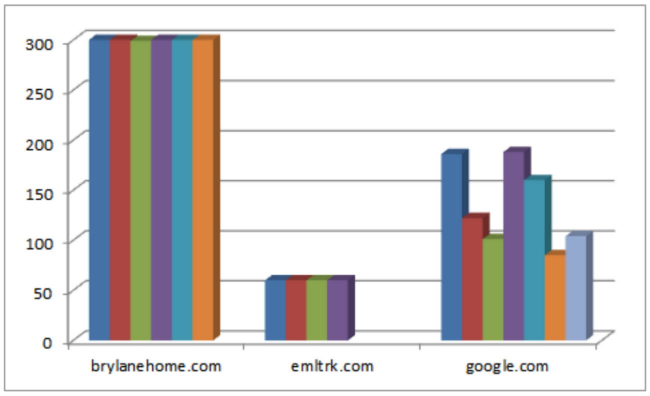


Fig. 9: TTL for Fast Flux and CDN domains.

2) Average Time-to-Live: To analyze the data obtained from the average time-to-Live we again segregate them into three sections, two for non-flux domains and another for fast flux domain.

There are websites which use large TTL value to resulting in the server caching their IP for hours. As can be observed in Figure 7, these set of observed domains display values in the range of thousands and ten thousands. The most observable abnormality would be the observed change in the TTL values of the IP address obtained for the domain "redtube.com". The implication of this difference will be discussed in a future section.

Most up to date websites use small values for TTL, at the expense of increasing the load on the server, as part of a Disaster Recovery Systems. Since these values are very small in the range of hundreds, they need to be observed separately from large TTL values to gain a better visualization (i.e., Figure 8 of the data.

Since CDN's resolve the same domain name to different IP addresses the Time-to-Live associated with the different IP addresses may show variations thus the need to observe them separately. As the TTL observed in Figure 9 for "google.com" domain shows variable value and needs further analysis.

When querying a DNS server for the resolution of a domain name, the query may return an error. This can be for many reasons, DNS is currently inactive, DNS refuses to resolve the domain query, the domain might be part of a blacklist referenced by the DNS, etc.Since a DNS server might reference local blacklist or global ones, the measured rate of error may prove to a good metric for conducting DNS analysis.

3) Observed Abnormalities: As mentioned earlier there were a few abnormalities observed in Figure 10. On conducting further analysis, the following data were obtained.

On further analysis of the IP resolutions and the associated TTL we noticed that the resolved IP's in some instances did not point to a Google domain. On conducting a reverse lookup the IP address (190.167.241.183), we found it pointed a host not associated with Google. This host was located in the Dominican Republic.

The abnormality observed in the TTL value for the "Redtube" domain was associated with the resolved IP (192.168.0.0

22) resolved using the DNS (202.52.255.47) located in Nepal. One can see that the resolved is a private IP range and therefore the resolution is definitely suspicious.

Though no significant abnormalities were observed for the "Facebook" domain the slight deviation if the TTL value warranted further analysis. The IP associated with suspicious behavior here is (10.10.34.0) which is also a private IP range. This was resolved using the DNS (2.179.65.58) located in Iran.

It is evident from the above results that the system can be used to conduct DNS based agility analysis.

VI. Related Works

There are a few existing works which we are utilizing to supplement our agility model to better predict the nature of a website. These references are segregated into three headers: (1) Lexical analysis of URL (2) Content Analysis and (3) DNS Analysis.

A. *Lexical Analysis*

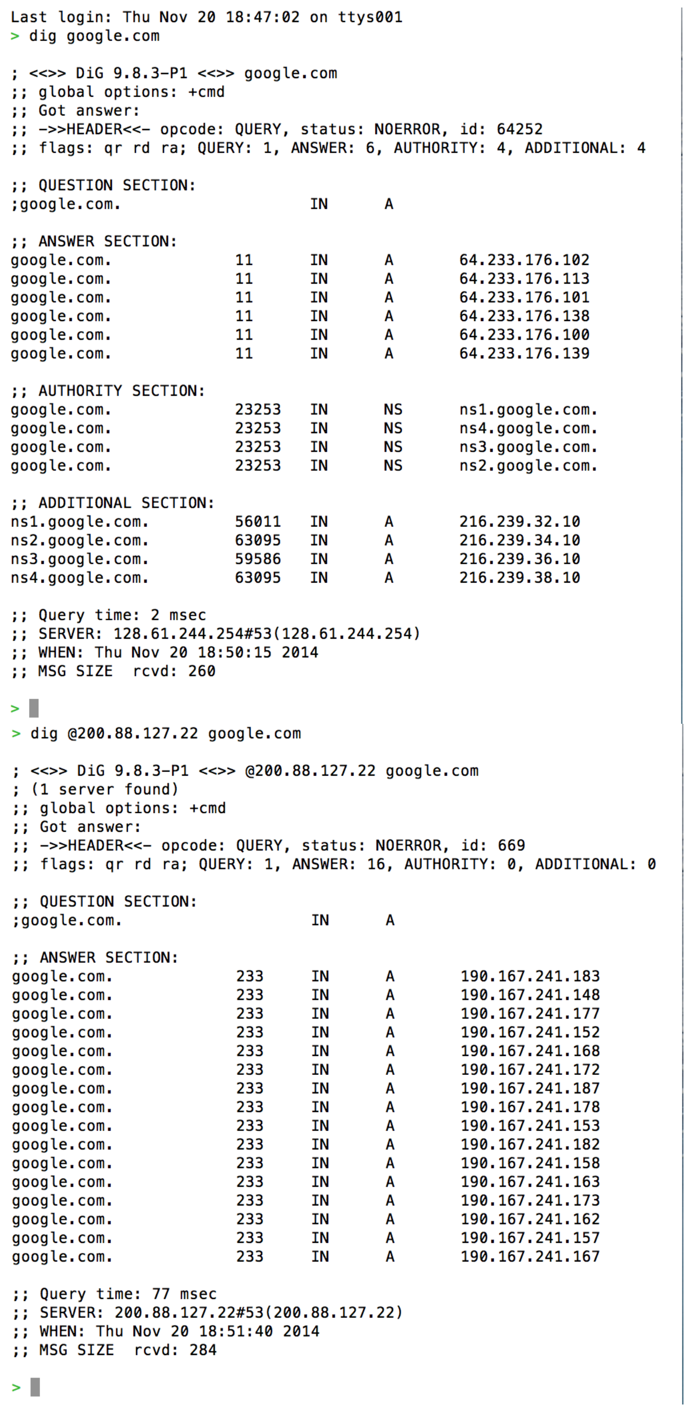
Lexical Analysis of a URL helps with the filtering of malicious websites and emails and in proactively tracking domain registrations to stop suspicious activity (spamming and phishing). There key characteristics which helps with the identification of a malicious website such as: (1) Anatomy of phishing URLs, (2) Domain name character composition, and (3) Presence of brands in URLs.

Our analysis is based on [3] and due to the related nature of our project we are deriving functionalities from their works to enhance our analysis. An earlier work [11] considers similar features and finds statistical differences such as distribution of domain name lengths between phishing and benign URLs. Their work, however, uses static analysis based on the data sets they have built while we have implemented a machine learning classifier to differentiate between benign and malicious URLs.

B. *Content Analysis*

Our aim is to perform web content analysis by storing and scrutinizing the following attributes in the content. The existence of iFrame is a good classifier on the malicious nature of a web page. We capture the content of the iFrames along with its attributes such as frame size, object tag count, embedded script tag count, JS functions and so on. The criteria and classifiers used to analyze the nature of a webpage (malicious or benign) has been derived from the works of Unmask Parasite [11] (a security blogging group), specifically from their work [6]. We were also inspired by [7], [15], as it was a great reference to guide us through our creative thought process.

[11] www.unmaskparasites.com



```
Last login: Thu Nov 20 18:47:02 on ttys001
> dig google.com

; <<>> DiG 9.8.3-P1 <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64252
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             11      IN      A       64.233.176.102
google.com.             11      IN      A       64.233.176.113
google.com.             11      IN      A       64.233.176.101
google.com.             11      IN      A       64.233.176.138
google.com.             11      IN      A       64.233.176.100
google.com.             11      IN      A       64.233.176.139

;; AUTHORITY SECTION:
google.com.             23253   IN      NS      ns1.google.com.
google.com.             23253   IN      NS      ns4.google.com.
google.com.             23253   IN      NS      ns3.google.com.
google.com.             23253   IN      NS      ns2.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.         56011   IN      A       216.239.32.10
ns2.google.com.         63095   IN      A       216.239.34.10
ns3.google.com.         59586   IN      A       216.239.36.10
ns4.google.com.         63095   IN      A       216.239.38.10

;; Query time: 2 msec
;; SERVER: 128.61.244.254#53(128.61.244.254)
;; WHEN: Thu Nov 20 18:50:15 2014
;; MSG SIZE  rcvd: 260

> 
> dig @200.88.127.22 google.com

; <<>> DiG 9.8.3-P1 <<>> @200.88.127.22 google.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 669
;; flags: qr rd ra; QUERY: 1, ANSWER: 16, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             233     IN      A       190.167.241.183
google.com.             233     IN      A       190.167.241.148
google.com.             233     IN      A       190.167.241.177
google.com.             233     IN      A       190.167.241.152
google.com.             233     IN      A       190.167.241.168
google.com.             233     IN      A       190.167.241.172
google.com.             233     IN      A       190.167.241.187
google.com.             233     IN      A       190.167.241.178
google.com.             233     IN      A       190.167.241.153
google.com.             233     IN      A       190.167.241.182
google.com.             233     IN      A       190.167.241.158
google.com.             233     IN      A       190.167.241.163
google.com.             233     IN      A       190.167.241.173
google.com.             233     IN      A       190.167.241.162
google.com.             233     IN      A       190.167.241.157
google.com.             233     IN      A       190.167.241.167

;; Query time: 77 msec
;; SERVER: 200.88.127.22#53(200.88.127.22)
;; WHEN: Thu Nov 20 18:51:40 2014
;; MSG SIZE  rcvd: 284

> 
```

Fig. 10: Showing DNS lookups with two different recursive DNS servers.

C. DNS Analysis

The Domain Name System (DNS) plays an important role in the operation of the Internet, providing a two-way mapping between domain names and their numerical identifiers. Given its fundamental role, it is not surprising that a wide variety of malicious activities involve the domain name service in one way or another. For example, bots resolve DNS names to locate their command and control servers, and spam mails contain URLs that link to domains that resolve to scam servers. Thus, it seems beneficial to monitor the use of the DNS system for signs of malicious activity. To this effect we derive our functionality inspired by two works, specifically the latter paper [1], [2].

VII. Conclusion

Determining if a website is benign or malicious is an important problem that has been studied by many researchers for quite some time. Lots of different analysis techniques have been studied and rendered effective to classify a website's true intent.

In the project, we described a comprehensive toolkit, Blackthread, that combines many effective analysis techniques used to identify the intentions of a website. Namely, our toolkit provides four types of analyses: (1) Website structure agility, (2) Lexical properties of URLs, (3) iFrame and JavaScript, and (4) DNS agility.

As shown, our classification efforts were successful and effective, with ROC curves with areas of up to 93% and 96%.

Future work includes extending this toolkit to other types of website analysis techniques and continuously collecting more ground truth to use to better the accuracy of labeling new websites in the future.

References

[1] A. Berger and W.N. Gansterer. Modeling DNS agility with DNSMap. In *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 387–392, April 2013.

[2] Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. Exposure: Finding malicious domains using passive dns analysis. In *NDSS*, 2011.

[3] Hyunsang Choi, Bin B Zhu, and Heejo Lee. Detecting malicious web links and identifying their attack types. In *Proceedings of the 2nd USENIX conference on Web application development*, pages 11–11. USENIX Association, 2011.

[4] Charles Curtsinger, Benjamin Livshits, Benjamin Zorn, and Christian Seifert. Zozzle: Low-overhead mostly static javascript malware detection. In *Proceedings of the Usenix Security Symposium*, 2011.

[5] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[6] Denis. Evolution of Hidden Iframes. blog.unmaskparasites.com/2009/10/28/evolution-of-hidden-iframes/, 2009.

[7] Yung-Tsung Hou, Yimeng Chang, Tsuhan Chen, Chi-Sung Laih, and Chia-Mei Chen. Malicious web content detection by machine learning. *Expert Systems with Applications*, 37(1):55–60, 2010.

[8] Alexandros Kapravelos, Yan Shoshitaishvili, Marco Cova, Christopher Kruegel, and Giovanni Vigna. Revolver: An automated approach to the detection of evasive web-based malware. In *USENIX Security*, pages 637–652, 2013.

[9] Marc Khrer, Christian Rossow, and Thorsten Holz. Paint it black: Evaluating the effectiveness of malware blacklists. In Angelos Stavrou, Herbert Bos, and Georgios Portokalidis, editors, *Research in Attacks, Intrusions and Defenses*, number 8688 in Lecture Notes in Computer Science, pages 1–21. Springer International Publishing, January 2014.

[10] Niels Provos Panayiotis Mavrommatis and Moheeb Abu Rajab Fabian Monrose. All your iframes point to us. In *17th USENIX Security Symposium*, pages 1–22, 2008.

[11] D Kevin McGrath and Minaxi Gupta. Behind phishing: An examination of phisher modi operandi. *LEET*, 8:4, 2008.

[12] Roberto Perdisci, Igino Corona, David Dagon, and Wenke Lee. Detecting malicious flux service networks through passive analysis of recursive dns traces. In *Computer Security Applications Conference, 2009. ACSAC'09. Annual*, pages 311–320. IEEE, 2009.

[13] C. Seifert, I. Welch, and P. Komisarczuk. Identification of malicious web pages with static heuristics. In *Telecommunication Networks and Applications Conference, 2008. ATNAC 2008. Australasian*, pages 91–96, December 2008.

[14] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. Shady paths: Leveraging surfing crowds to detect malicious web pages. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 133–144. ACM, 2013.

[15] Gang Wang, Jack W. Stokes, Cormac Herley, and David Felstead. Detecting malicious landing pages in malware distribution networks. In *DSN*, pages 1–11, 2013.