Building An Effective Malicious Domain Detector

Detection Of Malicious Domains With Select Features

Miguel Ranjo

Internet Services Final Paper

Internet Services

Rutgers University

United States

May 2022

1 Abstract

This paper is based off of Bilge et al.'s paper: *Exposure: A Passive DNS Analysis Service to Detect and Report Malicious Domains*, which I first came across as a related work in Shuaoi Hao and Haining Wang's paper: *Exploring Domain Name Based Features on the Effectiveness of DNS Caching* [3][7]. Malicious domains have become increasingly common and they pose a severe cybersecurity threat. Malicious domains look genuine but have been disguised by attackers to do things like steal sensitive, personal information, or data from unsuspecting victims. In this paper, we will choose select, well studied features for a malicious domain classifier in hopes that we can reduce critical weak spots in network security with minimal work. We will use empirical data and discuss the pros and cons of possible features extensively in this paper. For the features chosen, we focused on two things, choosing the minimum amount of features and having more passive features than other "active" papers. This idea came from Bilge et al.'s work and as a result of our analysis, we gathered data on Mean TTL Value, Lifetime of a Domain, Active Time of a Domain, and the Amount of Consecutive Characters. As a result, we end up with an f1-score of 81% for benign and 82% for malicious with only four features.

2 Introduction

Malicious domains have become more and more dangerous to the viability of the Internet over the past years. Threat actors abuse DNS and trick users through drive-by-download malware, botnets, phishing websites, and spam messages. Such attacks cause huge losses for individual and huge corporations and so, a means of detecting and classifying malicious domains would be very useful. Previously, past countermeasures against malicious domains were quite simple; you would filter domains based on blacklists. The issue is that this would not work on newly generated domains. Botnets in particular took this shortcoming into account, and they evolved to exploit this. One such example includes domain-fluxing which involves changing the domain name of the botnet's Command and Control server [4]. As a result, ML techniques started to be implemented to enhance detection capability in vetting domains.

The most primitive and common attacks for governments and civilians involve malicious URL's. The first thing we looked at is trying to identify malicious domains just by looking at URL's. The structure of a url looks like so:

https://www.example.com/index.html?a=paper

https is the protocol

example.com is the domain name, example is the SLD, and com is the TLD

index.html is the path

As mentioned before, a common means of finding malicious domains is to use ML models, but it is very easy for adversaries to use this info and circumvent these detection methods. For example, a very common ML feature is domain length which is easily circumvented by using a longer URL. Hence, we looked at multiple papers and their features, and landed on trying a select few features to build a better detection method.

3 Features

We look at various features mentioned among different papers, and we select a few that we see as most effective/least able to be circumvented. Some features were not implemented due to limitations on training time and hardware. Features implemented are marked with a *, but in summary, we implement mean TTL value, lifetime of a domain, active time of a domain, and consecutive characters.

3.1 Statistical

Number of IP Addresses Number of distinct IP addresses that appear in domain's DNS record. If the list appears as so ”198.126.8.10”, ”198.126.8.10”,”199.126.5.28”, then there are two distinct IP addresses. Implementation in other papers point to having more IP addresses as more desirable, but this is as easy for threat actors to circumvent by adding more IP addresses to the zone file [12].

Mean TTL Value* Average value of TTL was calculated by checking domain's DNS record 20 times. If we record a value of ”80” 15 times, and ”100” 5 times, we get a mean TTL of:

$$\frac{80*15+100*5}{20}=85 \tag{1}$$

Standard Deviation TTL Value Standard deviation is not robust compared to mean TTL value, hence standard deviation was completely avoided [6]. Looking at the SD equation:

$$s=\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i-\overline{x})^2} \tag{2}$$

It is clear that mean also influences SD, so it would make sense to include it [8]. On the contrary, it was found that setting DNS TTL values to 60,000 actually deceived the previous models and wrongly classified a URL as benign [6].

Lifetime of a Domain* This is the span between a domain's expiration date and creation date taken from Whois. For example, ”google.com” was updated on 09/09/2019, registered on 09/15/1997, and expires on 09/13/2028 [5]. Therefore, our lifespan is from 1997 to 2028 which is 30 years. This feature is very trustworthy as a threat actor would need to buy an old domain which is expensive if it even is feasible.

Active Time of Domain* Instead of lifetime which is the span between expiration and creation date, we look at active time which is the span between a domain's updated date and creation date. Looking at "google.com," we have a span from 09/15/1997 and 09/09/2019 which is 21 years [5]. Very similar to the lifetime of a domain, a threat actor would need to put a lot of resources in finding a domain with a particular active time hence this feature is robust.

3.2 Lexical

Domain Length The length of a domain is calculated by the domain name followed by the TLD. The minimum length of a domain is four since the domain name needs to be at least one character with most domain names having at least three. As stated before, domain length is incredibly unreliable as a measure because threat actors can simply increase their average domain length.

Consecutive Characters* In comparison, number of consecutive characters may seem easily manipulated, but due to the stark difference between benign and malicious when considering a low standard deviation, it is considered a robust feature [6].

Subdomain Malicious actors could mimic a legitimate site using a few subdomains. For example, an adversary could make google.g.com to mimic google.com. This feature would be pretty good in stopping phishing especially via emails, but it is not robust for this model as this is easily avoidable [15]. But this is a common mistake, so perhaps, a few of these features would be great for civilian-sided products.

Top Level Domain (TLD) Previous studies specifically point out that certain TLDs have historically been used for phishing urls [2]. This method is not robust as it is easy to avoid for threat actors.

Second Level Domain (SLD) As mentioned above, SLD refers to the google in google.com. There are a few ways to look at SLD which will be discussed in the next three features.

N-gram Papers use uni-gram, bi-gram, and tri-gram of the domain at character level [10]. We did not implement/analyze this feature due to its focus on"human nature", but we would like to consider it in a future update of the model.

Typosquatting Method Using the list of 500 top domains from Alexa (which are benign), it is possible to consider all type of "typos" including misspellings, singular versions, plural versions, hyphenations, and common domain extensions [1]. Alexa has unfortunately been discontinued as of May 1, 2002, so people who wish to use this method will have to use a different source (Alexa is a common data source for many of these papers) [3]. As a sample of typosquatting, go0ogle.com would be seen as close to google.com (with an 85% score) hence it is a typo [10]. The user would then be redirected to the correct site. This is very similar to Google buying every domain name similar to theirs, but a lot less expensive (look up gooogle.com and other longer o domains on Whois) [5]. Just like n-gram, it focuses on human typos, and as a result was not implemented.

Entropy Entropy is like an expansion on consecutive characters where it looks at repetition at the whole domain level. This feature is easily manipulated to line up within the parameters of other papers (such as keeping entropy less than 4) [16]. As a result, this feature is not robust and not implemented.

4 Training Set

The quality of the results depends greatly on the quality of the training set. With this in mind for malicious domains, classification algorithms tend to be powerless due to the imbalances in training data sets. Specifically, malicious domains are relatively rare in DNS traffic data as 99% is devoted to benign [10]. As a result, we looked through multiple data sets including the CIC-Bell DNS 2021 data set, Mendeley Data benign and malicious data set, DITL data sets, and the CAIDA data set. Relevant data sets are described below.

The CIC-Bell data set includes 400,000 benign and 13,011 malicious samples from 1,000,000 benign and 51,453 malicious domains garnered from publicly available datasets.

The span of this set is from May 2019 to June 2019. This set was separated into spam, phishing, and general malware [10]. It is also the main dataset we pulled our domains from. We pulled every single domain they looked at (1,000,000 benign and 51,453 malicious) and formed our own data. More about that in the implementation section.

Mendeley Data uses public DNS logs and actually includes a lot of the features we talk about above. Among these features, there is even geolocation features which will be talked about in future research. This dataset contain 90,000 domains balanced equally between malicious and benign [13].

The CAIDA data set was useful in providing daily and quarterly files with TTL times. These sets were collected from CAIDA's ark, but the data is old (April 2014). I mainly used this set to look at TTL times to understand what the average would look like among such a large sample (as I was unsure if my results on mean TTL time were correct) [9].

5 Implementation

From the data set, we have to modify a few things. In particular, the data sets needed to be updated with data for all our features. We use Python packages like *whois*. Whois is a very helpful website for domains, and the *whois* package is used to grab updated dates, creation dates, and expiration dates. Differences between these dates become "Active Time of Domain" and "Lifetime of Domain" which is directly used in our model. Whois also helps us take malicious and benign URLs to our URL data set before it is transferred to our DNS data set. We use *OS* to using the *ping* command 20 times and pipe it back to calculate the mean TTL time. Data sets are updated with these numbers before being put into the classifier. To note, due to the size of the data, we divided the data set into bite-sized pieces for the code to run through. If any of the feature values were missing, we did not include it into the final data set. With this in mind, we saw spam numbers reduced from 8,254 to 190; phishing completely eliminated (which comes from not being registered in Whois and with it, average TTL was not calculated); malware reduced from 51,453 to 2,267; and benign reduced to 21,066. There was no shortage of potential benign domains to pull from, but to note, we did not need all the benign domains we could have added. Finally, consecutive characters was very simple to implement and we take the highest number of consecutive characters as the score for domains. This data set was put into a classification model and split into benign and malicious from which each set's features were analyzed.

6 Discussion/Results

There were a few fascinating stats from each data set. We thought that excluding the number of consecutive characters would not make a huge difference in our model (we did not believe the result of previous papers), but after considering this feature, we saw our model become more accurate. This is not to say that because it made our model more accurate we should include it (as we included it because of past papers), but it was just an interesting observation made while doing our analysis.

We split the results into benign and malicious means and standard deviation. Here, we found the benign mean for consecutive characters to be 1.05 with an SD of 0.24 and the malicious mean to be 1.08 with an SD of 0.33. Mean TTL value's benign mean was 72.68 with an SD of 50.99, and the malicious mean was 69.79 with an SD of 42.23. Lifetime of domain's benign mean was 14.65 years and SD was 8.43 years and malicious mean was 8.92 years and SD was 6.41 years. Active time of domain's benign mean was 12.62 years and SD was 7.80 years and malicious mean was 7.36 years and SD was 5.98 years. All these results lie within expectation.

I, in particular, am not satisfied with the features chosen. The three statistics we looked at, mean TTL, lifetime, and active time have been identified as robust in papers but lack a preferred flexibility. This difference is much like a comparison between accuracy and precision. The reason these three statistics work is due to the investment made by companies. Either they were in the scene early enough and kept the same domain, bought an old domain for an exuberant amount, or they invested a lot into infrastructure. By infrastructure, I refer to the mean TTL. The longer the TTL, the more likely the domain is running on one server that does not frequently change their IP configuration. Meanwhile, short TTLs are useful for domains that change their records frequently, but require more servers, a load balancer, and even rules set by ISPs for minimum allowable DNS refresh rates. It would be great to implement more features that are more useful for a decentralized network instead of these statistics that prefer established domains close to big money. An additional smaller issue is the availability of such data. Compared to the other datasets, we filtered a lot of potential benign and malicious candidates due to which data we chose to collect.

After the model was trained and parameters adjusted, we achieved a test and train accuracy of 90%. Looking further into the f1-score of benign, we get a 95% and a f1-score of malicious of 1%. Looking back at the data, the above metrics line up with 2,267 malicious and 21,066 benign or only 9.8% of the data being malicious. We see this issue in which we get an amazing showing for benign, but a horrible one for malicious. To improve this, we reduce the amount of benign examples. Next, we reduce the benign size to 9,000 or 20% which gives us an accuracy of 80%. This yields an f1-score of 89% on benign and 5% on malicious. Finally, we try 50%, an equal divide. As expected, it gave us our best results, an f1-score of 80% for both benign and malicious with an 80% accuracy. It seems that a huge gate to making a great classifier is having enough malicious data. Perhaps as more malicious websites come to light and data is collected, we can further improve the classifier with these features. We can add our extra spam domains and we end up getting a final f1-score of 81% for benign and 82% for malicious with an 82% accuracy.

7 Future Research

As mentioned before, a huge worry about this method is its robustness. If threat actors start considering the analysis/data included here and other papers, they can easily adjust their methods to circumvent detection. Although this paper did take that into account and chose specific methods to prevent this, the field/statistics could change as more adversaries implement certain features. With that in mind, it seems that the two ”times” of domains are the most robust, but also the most inflexible. If a civilian or company wants to open up a site, they would not pass the lifetime and active domain tests easily and would have to depend on other features. Now if they had the money, they could get to the Goldilocks zone for mean TTL. It would also be interesting to see more features implemented from other papers especially n-gram. Perhaps this would increase the accuracy and the flexibility of our testing method.

Another method mentioned in papers is analyzing geolocation including location of organization’s owner, country of server, and amount of unique countries. Some papers implement features like Communication Countries Rank which ranks specific URLs based on country [6]. The number of passive DNS changes is looked at as benign domains have much more significant DNS changes (26.4 vs 8.01) [2].

Additionally, I would like to look at the other side of these papers and act as a threat actor trying to circumvent their detection algorithms. In particular, I want to see if the features I chose stand the test of time, which means although they are less accurate, they may be better used in the long term. Such research can further the security of the models and make them more effective.

With consideration of usability, perhaps it would be more worthwhile to implement a ”likeliness” of a domain being malicious especially if a domain appears in an email. Here, n-gram and other lexical features are perfect. It would be helpful for civilians as it would make them wary of putting any private info on sites (or clicking on them in the first place). However in the continuous game of cat and mouse between bad actors and everyone else, it could just escalate the game as it provides them a baseline for which to perform on the easiest stage (against ”non-technical” opponents). Then again, for this case, it seems very hard to revolutionize the field.

8 Conclusion

Although we implemented a successful means of finding malicious domains, the field will evolve as time goes, and these features will become outdated in the future. As mentioned before, there is always going to be a game of cat and mouse where the bad actors have the upper hand as they exploit and everyone else has to react to them. Even if we create a reactive format that can quickly block them, the internet is about being an open network to communicate, so you may end up ”effective” but inflexible as you block a ton of benign domains. Such a means like only marking older domains as benign winds up defeating the purpose of the internet. We will have to look for more robust features to add to the arsenal that this our detection model. This, however, is a great baseline to start. At the end of the day, the best method of dodging bad actors is to have great internet hygiene and to stay aware: know what you click/where you go to, do not download from untrusted sources, and of course, use strong passwords and do not reuse them.

References

[1] Alexa Top Sites. (n.d.). Retrieved April 16, 2022, from https://www.alexa.com/topsites

[2] Bao, Z., Wang, W., & Lan, Y. (2019). Using passive DNS to detect malicious domain name. Proceedings of the 3rd International Conference on Vision, Image and Signal Processing. https://doi.org/10.1145/3387168.3387236

[3] Bilge, L., Sen, S., Balzarotti, D., Kirda, E., & Kruegel, C. (2014). Exposure. ACM Transactions on Information and System Security, 16(4), 1–28. https://doi.org/10.1145/2584679

[4] Choi, H., Lee, H., Lee, H., & Kim, H. (2007). Botnet detection by Monitoring Group Activities in DNS traffic. 7th IEEE International Conference on Computer and Information Technology (CIT 2007). https://doi.org/10.1109/cit.2007.90

[5] Every great idea starts with a great domain name. whois.com. (n.d.). Retrieved May 3, 2022, from https://www.whois.com/

[6] Hajaj, C., Hason, N., & Dvir, A. (2022). Less is more: Robust and novel features for malicious domain detection. Electronics, 11(6), 969. https://doi.org/10.3390/electronics11060969

[7] Hao, S., & Wang, H. (2017). Exploring domain name based features on the effectiveness of DNS caching. ACM SIGCOMM Computer Communication Review, 47(1), 36–42. https://doi.org/10.1145/3041027.3041032

[8] Hao, S., Feamster, N., & Pandrangi, R. (2011). Monitoring the initial DNS behavior of malicious domains. Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference - IMC ’11. https://doi.org/10.1145/2068816.2068842

[9] Index of /datasets/topology/ark. (n.d.). Retrieved May 2, 2022, from https://publicdata.caida.org/datasets/topology/ark/

[10] Ipinfo. (n.d.). 205.174.165.80. IPinfo. Retrieved May 2, 2022, from https://ipinfo.io/205.174.165.80

[11] ISI Analysis of Network Traffic Lab. (n.d.). Datasets about DNS. Retrieved May 2, 2022, from https://ant.isi.edu/datasets/dns/

[12] Liu, Z., Zeng, Y., Zhang, P., Xue, J., Zhang, J., & Liu, J. (2018). An imbalanced malicious domains detection method based on passive DNS traffic analysis. Security and Communication Networks, 2018, 1–7. https://doi.org/10.1155/2018/6510381

[13] Marques, C. (2021, June 9). Benign and malicious domains based on DNS logs. Mendeley Data. Retrieved May 2, 2022, from https://data.mendeley.com/datasets/623sshkdrz/5

[14] Moura, G. C., Heidemann, J., Schmidt, R. de, &amp; Hardaker, W. (2019). Cache me if you can. Proceedings of the Internet Measurement Conference. https://doi.org/10.1145/3355369.3355568

[15] Palaniappan, G., S, S., Rajendran, B., Sanjay, Goyal, S., & B S, B. (2020). Malicious domain detection using machine learning on domain name features, host-based features and web-based features. Procedia Computer Science, 171, 654–661. https://doi.org/10.1016/j.procs.2020.04.071

[16] Zhauniarovich, Y., Khalil, I., Yu, T., & Dacier, M. (2019). A survey on malicious domains detection through DNS data analysis. ACM Computing Surveys, 51(4), 1–36. https://doi.org/10.1145/3191329