Frequency analysis of second-level domain names and detection of pseudo-random domain generation

Ryan Doyle 12 June 2010
rd@ryandoyle.net

Abstract – This paper will analyse the letter distribution found in second level domains for the .net, .com and .org top level domains. This distribution will be compared with the letter distribution of pseudo-randomly generated domains found in malware. An investigation into the possibility of detecting randomly generated domains solely using frequency analysis will be proposed as well as a system to detect infected computers querying a Domain Name System server.

1. Introduction

The Domain Name System is an important part of the Internet's architecture. It provides distributed and fault tolerant methods for an Internet connected user to lookup services provided by other Internet connected devices using English[*] characters. It can therefore be assumed that the distribution of letters contained in common domain names also conforms to similar distributions found in the English language. This paper asks the question – *With what certainty can we detect pseudo-randomly generated domain names such as found in Internet worms like Conficker using only frequency analysis?*

2. Frequency analysis of second-level domains

Frequency analysis is a tool to analyse the density of characters found in text[2]. It can be used to break simple letter shift cryptography and prove or disprove authenticity of a written text back to an author just to name a few. This paper proposes using frequency analysis as a tool to detect randomly generated domain names.

Test conditions

Logs from the primary internal DNS server for a K12 school in Australia were used as the source of the frequency analysis. The DNS server was running Windows Server 2003R2, using the Microsoft DNS server. The following data was collected between the days of 14th– 18th October 2009, constituting a full work week. The school has approximately 500 students and 100 staff, both administrative and teaching. This brings the potential user base to 600 users. DNS queries mainly include those generated by web browsing.

[*] With the use of Unicode support for the Domain Name System, this can now include non US-ASCII characters.

How logs were analysed

Logs were broken down into searching for only A records. Any level of the domain past the second-level, (EG: `stripped.example.com`) was removed. Only unique domain names were stored preventing any duplicates. For example, `www.example.com` and `mail.example.com` were only stored once as `example.com`. Over 11 million lines of DNS logs were processed in the week. This equated to 126,694 individual A record queries. This was broken down into 3381 unique second-level domain names. Only `.com`, `.net` and `.org` top-level domains were analysed. Custom written tools were used to analyse the domain names and create a frequency table of the letter distribution[6].

Findings

The letter frequency of second-level domain names was found to be similar to reference frequency tables[1,3]. The reference frequency tables are a generic frequency table for the English written word. The density was calculated by the number of times a letter appeared divided by the total number of letters analysed.

	Ref.	**Sample**		**Ref.**	**Sample**
a	0.08167	0.08673	***n***	0.06749	0.06246
b	0.01492	0.02242	***o***	0.07507	0.07256
c	0.02782	0.04300	***p***	0.01929	0.02890
d	0.04253	0.03547	***q***	0.00095	0.00203
e	0.12702	0.10543	***r***	0.05987	0.06705
f	0.02228	0.01595	***s***	0.06327	0.07062
g	0.02015	0.02896	***t***	0.09056	0.06821
h	0.06094	0.02371	***u***	0.02758	0.02965
i	0.06966	0.07400	***v***	0.00978	0.01376
j	0.00153	0.00344	***w***	0.02360	0.01667
k	0.00772	0.01473	***x***	0.00150	0.00588
l	0.04025	0.04898	***y***	0.01974	0.01867
m	0.02406	0.03569	***z***	0.00074	0.00504

Table 1 – Reference frequency table with the second-level domain frequency data densities

It is to be expected that domain name frequency tables would be similar to the reference frequency table as many domain names follow English nouns, verbs, adjectives or combinations of these. There are some significant differences such as “h” and a fairly large reduction on “e” and “t” but for the most part, the frequency follows a similar trend.

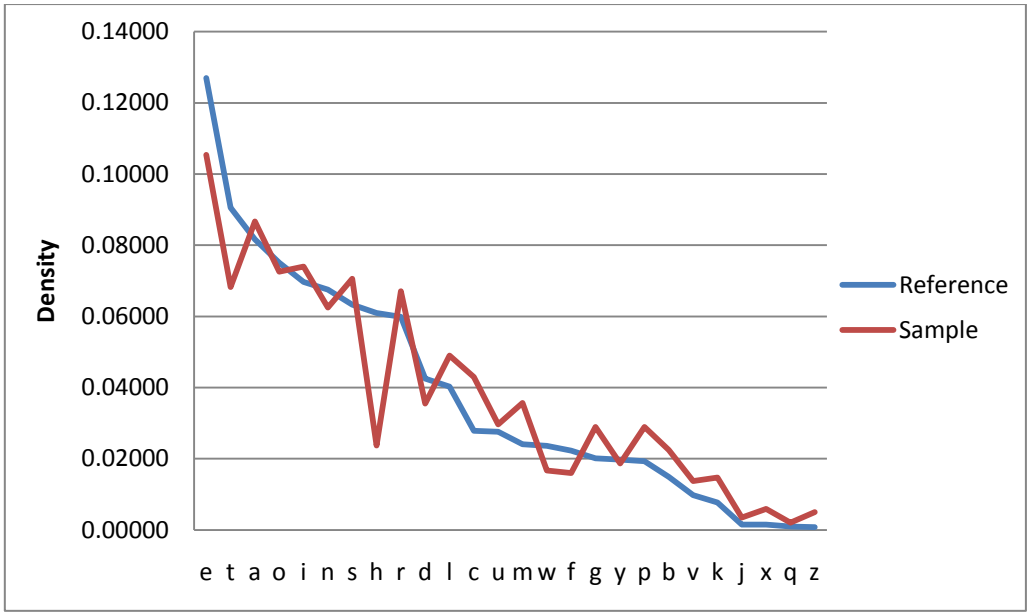

Figure 1 – Letter frequency of the reference versus the second-level domain sample.

For what discrepancy there is in Figure 1, this could be the frequency distribution table for domain names simply does not follow a general frequency distribution table. Different frequency tables exist for different uses of the English written word[1] so it is not uncommon to see the discrepancy in the data. Note that the sample data looks a lot more erratic compared to the reference. This is only because we are ordering the x-axis as per the density of the reference frequency. Although the graph is represented as a line graph, the lines are only used as an indication as to the proximity of each data source to one another. There are no arbitrary values between the data points on the x axis.

3. Frequency analysis of pseudo-randomly generated second-level domains

The Conficker worm was used as an analysis for pseudo-random generated domains. The same size sample of 3381 domains was selected off a publically available list[5] of over 1 million Conficker domains. The same tools were used to generate the frequency table as the legitimate domain names in the previous section.

Findings

The letter frequency for the Conficker domains was *much* more uniform than the reference frequency. This is to be expected of the pseudo-random algorithm as any good pseudo-random algorithm should give us an even distribution within its domain[†].

	Ref.	Conficker		Ref.	Conficker
a	0.08167	0.03821	***n***	0.06749	0.03955
b	0.01492	0.03829	***o***	0.07507	0.04018
c	0.02782	0.03725	***p***	0.01929	0.03911
d	0.04253	0.03781	***q***	0.00095	0.03940
e	0.12702	0.03885	***r***	0.05987	0.03810

[†] Domain in this instance is referred to as the possible values that could be output of the pseudo-random generating algorithm

f	0.02228	0.03918	***s***	0.06327	0.04000
g	0.02015	0.03677	***t***	0.09056	0.03588
h	0.06094	0.03799	***u***	0.02758	0.03751
i	0.06966	0.03840	***v***	0.00978	0.03903
j	0.00153	0.03788	***w***	0.02360	0.03655
k	0.00772	0.03755	***x***	0.00150	0.03784
l	0.04025	0.04111	***y***	0.01974	0.03888
m	0.02406	0.03744	***z***	0.00074	0.04126

Table 2 – The Conficker frequency table shows a much more stable distribution.

These results may seem fairly obvious but that does not mean the implications are not important. The linearity of the frequency can be visualised when is compared with the reference frequency in Figure 2.

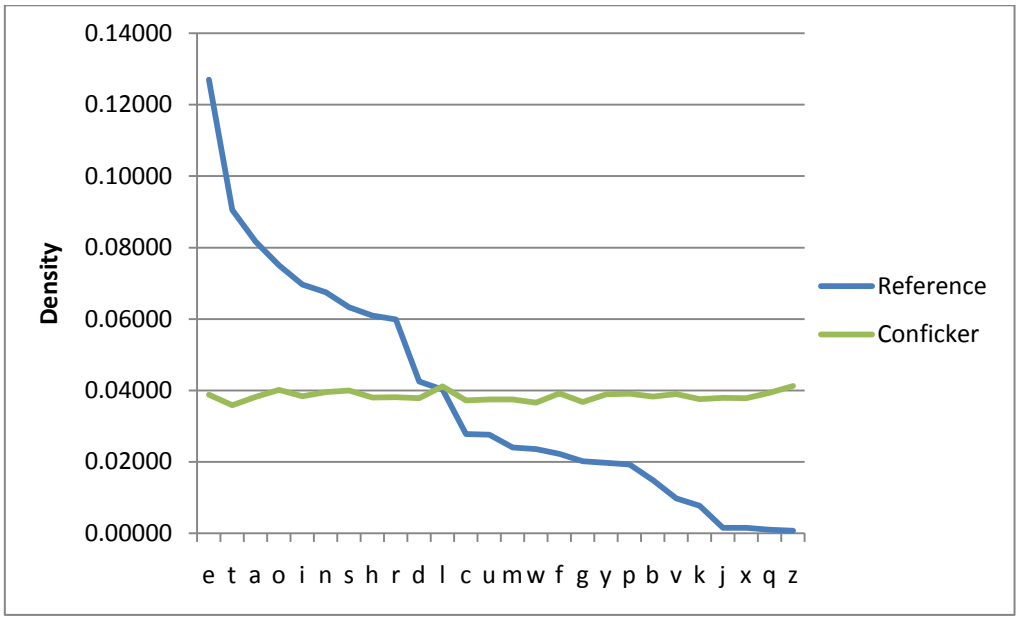

Figure 2 – The Conficker domains letter frequency is almost linear.

For all intensive purposes the Conficker domains letter frequency can be considered linear. A completely even distribution would be a flat line at 1/26 or about 0.0385. Additional frequency analysis was performed on a larger sample of Conficker domains of over 1 million, which produced an even greater linearity.

4. Combining Conficker and real-world data sets

The real-world frequency table will be assumed as the “correct” frequency table for domain names. Frequency tables differ for different uses of English, so it is not uncommon to see small variations between the real-world domains and the reference as shown in Figure 1.

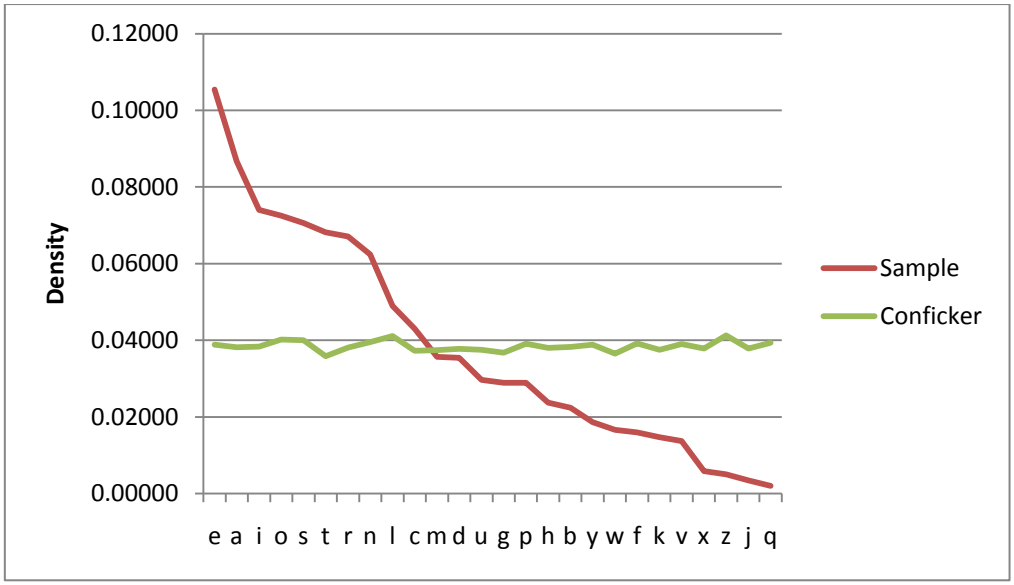

Figure 3 – The real world sample domains shown with the Conficker domains

Figure 3 shows a similar trend as previously demonstrated in Figure 2. The only difference now is that the real-world sample domains are assumed the correct frequency.

Weighted score-based analysis

The two distinct distributions allow for additional analysis of a score-based system for pseudo-domain detection. A simple weighting formula was devised to be able to plot a cumulative distribution of a weighted letter frequency.

$$w = \frac{\sum_{i=0}^{n} x_i}{n} \times 1000$$

$$Where:$$
$$w = weighting$$
$$n = num.letters\ in\ domain$$
$$x = frequency\ lookup\ of\ letter\ i$$

The frequencies from the Sample column in Table 1 are added up for each character in the domain name, excluding the top level domain (EG: `.com`) and then divided by the number of characters in the domain. The result is multiplied by 1000 to give a weighting easier to work with. The resultant weighting will follow the limits presented below.

$$0 < w < 106$$

The limit of 106 is calculated by multiplying the highest frequency letter “e” by 1000 and applying a ceiling function. All scores using the weighted letter frequency function will fall within these limits when using the frequency distribution table of the sample domains presented in this paper. For an example, the domain `google.com` will get the score of 59.575. This is calculated by:

$$(.02896_g + .07256_o + .07256_o + .02896_g + .04896_l + .10543_e) \div 6 \times 1000$$

The weighted letter frequency formula was applied to the list of real-world sample domains as well as pseudo-random Conficker domains. The cumulative weighting scores can be seen in Figure 4.

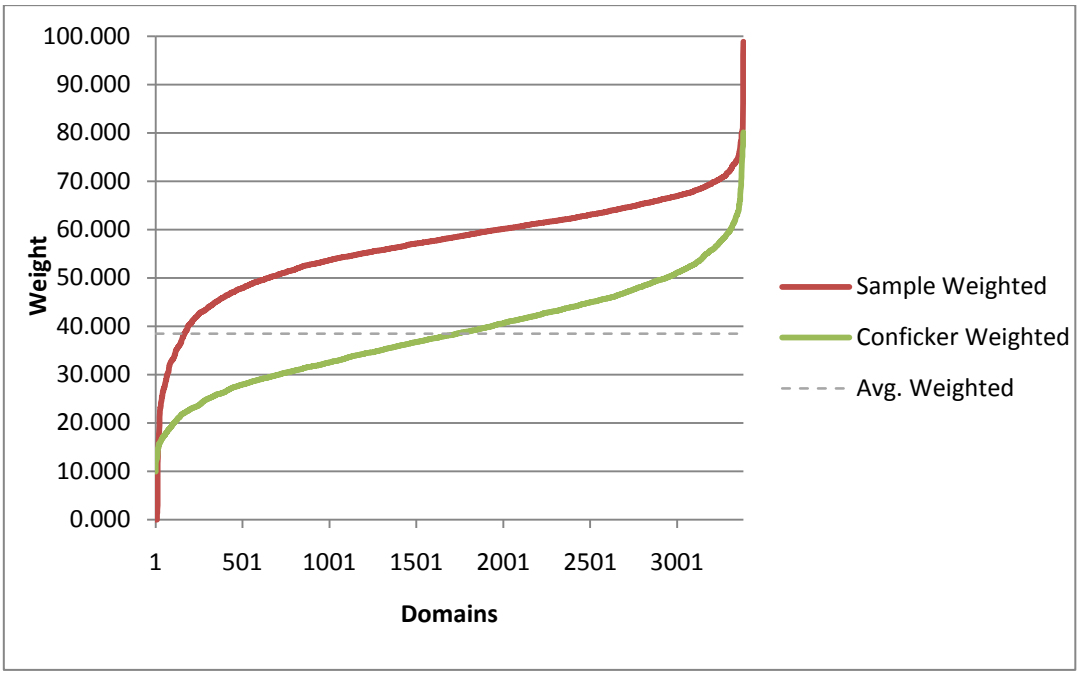

Fig 4 – Cumulative weightings of real-world sample domains and pseudo-random Conficker domains

Using the plot and the data that generated the plot, some basic under-blocking versus over-blocking predictions can be determined. The following conditions were proposed to apply these over-blocking versus under-blocking results.

a. Block 80% pseudo-random domains and determine the percentage of false positives
b. Block 95% pseudo-random domains and determine the percentage of false positives
c. Achieve no more than 20% false positive and determine the percentage of pseudo-random domains that would be blocked.
d. Achieve no more than 5% false positive and determine the percentage of pseudo-random domains that would be blocked.

In example *a*, 80% blocking of pseudo-random domains would require a score of < 46.994 to be blocked. When we compare this to the real-world sample scores, we would only over-block these domains by 13.07%. That is fairly impressive using *only* frequency analysis and nothing else.

Example *b* is just an extension of example a, but with an increased requirement for pseudo-random domain detection of 95%. A score of < 56.001 would be blocked in this instance. This achieves a considerable amount of over-blocking the real-world sample domains by 39.49%.

Example *c*, the limits are set on false positives of the real-world sample domains. In this example, the limit is set to no more that 20% false positive. Note that this is fairly high as false positives go for real world applications. A score of < 50.345 would be blocked in this scenario. This equates to a 87.55% blocking rate for pseudo-random domains.

Similarly, in example d, a more respectable false positive rate it set at 5%. In this example, a score of < 38.587 would be blocked. This equates to a blocking rate of 51.73% for pseudo-random domains.

This is quite an acceptable figure. For the sake of 5% false positive, a blocking rate of 51.73% is fairly good given that the only information is the domain name itself.

5. Limitations

As impressive as the results can be, given the small amount of information that we know, there are several limitations.

- Not all malicious domains are going to be random letters that follow an even distribution
- There is nothing stopping a worm integrating the frequency table listed in Table 1 as a factor of bias to the pseudo-random generation algorithm. Even forcing all domains to include the letter "e" could dramatically increase the chances the domain name would go undetected.
- False positives are going to happen

Knowing these limitations, it is important to build applications that use the weighted score to compliment other elements that determine a domain's validity.

6. Applications

With false positives quite a possibility, it would not be recommended to build a system that used frequency analysis alone to determine the validity of a domain name. The weighting determined from the weighting formula should contribute to an overall rating as to the validity of the domain. Much like SpamAssassin[8] determines legitimate email from spam by giving it a score using many different factors, a domain name should be given similar treatment.

We could detect an "attack" on a DNS server by seeing a shift in frequency distribution to becoming more linear. With only 3381 legitimate domains albeit only .com, .net and .org were selected, it could be assumed that this value would be no larger than double, 6792 to include all other top level domains. Compare this to the 50,000[7]potential domains that would be queried if a machine was infected with Conficker, we would see a fairly dramatic shift linearity quickly.

7. References

[1] Lewand , Robert E. "Cryptological Mathematics", The Mathematical Association of America, 2000, [Online]. Available: http://books.google.com/books?id=CyCcRAm7eQMC&pg=PA36

[2] Letter frequency, http://en.wikipedia.org/wiki/Letter_frequency, March 2010

[3] Statistical Distributions of English Text, http://www.data-compression.com/english.html, March 2010

[4] P. Porras, H. Saidi, V. Yegneswaran, "Conficker C P2P Protocol and Implementation". [Online]. Available: http://mtc.sri.com/Conficker/P2P/index.html

[5] Conficker Domain Information, http://blogs.technet.com/msrc/archive/2009/02/12/conficker-domain-information.aspx, March 2010

[6] R. Doyle, "DNS Utils", http://ryandoyle.net/devel/dnsutils/, March 2010

[7] F. Leder, T. Werner, "Know your enemy: Containing Conficker". [Online]. Available: https://www.honeynet.org/files/KYE-Conficker.pdf

[8] The Apache SpamAssassin Project, http://spamassassin.apache.org/, March 2010