Proceedings of the Singapore Cyber-Security Conference (SG-CRC) 2016
A. Mathur and A. Roychoudhury (Eds.)

doi:10.3233/978-1-61499-617-0-139

Accurate in-network file-type classification

Dinil Mon Divakaran, Yung Siang Liau, and Vrizlynn L. L. Thing
*A*STAR Institute for Infocomm Research (I^2R), Singapore*

Abstract. Accurate classification of file types carried by network traffic aids in securing a network against various types of malicious activities such as malware infection, data exfiltration, botnet communication, etc. An important challenge here is to accurately classify files without slowing down network traffic. Therefore, the cost of accurate file-type classification has to be known. In this work, we carry out a preliminary but extensive investigation to evaluate different sets of features for file-type classification. The objective is to detect not only file types under normal scenario, but also files that are transferred with obfuscated headers. Our experiments show that the feature vector consisting of unigram frequencies leads to high accuracy; yet, combining this feature set with entropy feature vector leads to improvement in accuracies.

Keywords. Classification, files, network, real-time, n-grams, entropy

1. Introduction

Network traffic is an important medium for security breaches, and therefore for security analysis. Exfiltration of sensitive data from an enterprise, download of malicious executables, covert communications, sharing of copyrighted software, etc. happen over a network. In this work, we conduct preliminary research to detect the types of contents or files carried by traffic flows. Real-time content classification aids in securing a network, be it an enterprise network, an ISP network or a government organization. For example, a financial enterprise often blocks encrypted attachments in mails, so that the administration can monitor the contents of email exchanges. ISPs are interested in blocking malware (which are typically in binary format) before they reach end-users. Similarly, a government organization engaging in tenders might want to scrutinize files before they leave the network. Yet, another organization network, like that of a retail store, would not expect any kind of binary executables or codes in their traffic. Identifying file types is also required in forensic analysis, for example, to detect file fragments and recover files from computers and mobile systems [1].

While accuracy of file-type classification is paramount to its utility, the efficiency of classification cannot be ignored—such classification solutions are typically deployed at the perimeters of a network. Routers at the edges of a network are connected using high capacity of links, and depending on the network load, a router sees large numbers of flows and packets traversing through it. Given this realistic scenario of deployment, it is necessary to consider the cost of classifying

file types carried by network traffic. Cost is in terms of, both, the number of bytes or packets that need to be buffered, and the computational runtime required for feature extraction (and classification).

In [2], the authors study a number of features (and their combinations) such as n-grams, distribution frequencies as well as Shannon's entropy and Kolmogorov complexity, for classifying file types. While the evaluation is done for 30 file types, fragments of 512 bytes (for both training and testing) are extracted from a small number of files—2,587. This naturally makes sampled fragments dependent, potentially affecting results. Overall accuracy of 73.4% is reported, using a combination of unigram and bigram frequencies. But feature vector of bigram frequencies has a dimension of 65,536, making it impractical for real-time classifications.

In a recent research work [3], block entropies (refer Section 2.3) are used for classification of files in network traffic. The work compares results of classification using the byte-streams of variable sizes from both the beginning of file and random locations. We compare block entropies in our evaluations as well (using the same feature vector as in [3]). From our experiments, we observe that block entropies (which require higher computational time) are not required for accurate classification of files using the beginning of the file; indeed unigram frequencies feature vector alone gives high accuracy in this scenario. For streams extracted from random locations, a combination of feature vectors (including block entropies) perform better than a feature vector consisting of only block entropies.

In this paper, we differentiate features based on the costs, and evaluate the accuracy of classification using different feature sets and their combinations, while varying the number of bytes required for classifying a flow. We describe the different feature vectors used for classification in the following section. Subsequently, we evaluate the goodness of different feature vectors in classifying file types, using a database of around 60,000 files and nine file types.

2. Feature vectors for classifying file types

2.1. n-gram frequencies

Given a stream of bytes, a commonly used set of features is the n-gram frequencies [2,4,5]. In case of unigram, the frequencies of bytes are stored in a vector of dimension 256. The size of the feature vector increases exponentially with increasing size n-grams (or n), and so does storage space required for computation of the feature vector. Besides, our experiments have shown that bigrams and higher orders do not bring in additional value in comparison to unigram frequencies for file-type classification. Therefore, we use a feature vector consisting of only unigram frequencies in this work. The computation of unigram frequencies is straight-forward—for a byte-stream of size m, a single pass over the m bytes gives the frequencies of unigrams.

2.2. Moments of the probability distributions

The feature vector consists of mean, standard deviation, skewness and kurtosis. Skewness for a stream of bytes is a measure of symmetry of distribution of bytes.

For sample values, the skewness is calculated as, $b_1 = \frac{1}{m}\frac{\sum_{i=1}^{m}(x_i-\bar{x})^3}{\bar{\sigma}^3}$, with $\bar{x}$ and $\bar{\sigma}$ being the mean and standard deviation, respectively, of the m samples.

The fourth measure is kurtosis, which characterises the peakedness of a distribution. The kurtosis of m samples, $k = \frac{1}{m}\frac{\sum_{i=1}^{m}(x_i-\bar{x})^4}{\bar{\sigma}^4}$. While still capturing the characteristics of the byte distribution in a given stream similar to unigram frequenies, the feature vector here has a much small dimension of four.

2.3. Block entropy

Entropy a measure of randomness of a random variable. For a discrete random variable X taking values in $\mathcal{X}$, entropy is defined as: $H(X) = -\sum_{x\in\mathcal{X}} p(x)\log(x)$, where p is the probability mass function. Following conventions, we take $0\log 0 = 0$. We measure entropy in bits, i.e., the log is taken to the base 2.

For contents in a given file, $\mathcal{X}$ is the set of all values a byte takes. However, the above computation of entropy can be generalized for byte-blocks as well. For example, when we consider a block of byte one, $|\mathcal{X}^1| = 2^8$. For blocks (or equivalently, sequences) of two bytes, $|\mathcal{X}^2| = 2^{8*2}$. Hence, we can generalize the entropy computation over a block of i bytes as: $H_i(X) = -\sum_{x\in\mathcal{X}^i} p(x)\log(x)$.

Block entropies were used in [3] for file-type classification. The complexity of computing block entropy is $\mathcal{O}(m^2)$ for a stream of m blocks.

2.4. Entropy of data from Markov information source

When the data source has a Markovian property (present is dependent on the past), as is the case with text documents, the probability of a byte in a sequence is conditional on the probabilities of occurrences of the preceding i bytes. Here i is referred to the order or the memory that needs to be recorded. For a Markov information source of order $i-1$, entropy is computed as follows;

$$H_i^M = \sum_{x_1\in\mathcal{X}} p(x_1) \sum_{x_2\in\mathcal{X}} p(x_2|x_1)... \sum_{x_i\in\mathcal{X}} p(x_i|x_1x_2...x_{i-1}) \log p(x_i|x_1x_2...x_{i-1}) \tag{1}$$

The runtime complexity of computing H_i^M is $\mathcal{O}(m^i)$ for order $i-1$, where m is the number of bytes in the stream.

3. Performance evaluation

This section presents results form the experiments conducted. Before proceeding, we discuss on the data used for evaluation purposes, and the experiment settings.

3.1. Data

We collected around 60,000 files for training and testing purposes. The different file types forming the data are: encrypted (*enc*), executables (*exe*), PDF (*pdf*), JPG (*jpg*), PNG (*png*), zip (*zip*), gzip (*gz*), MP3 (*mp3*), plain text and HTML.

Plain text and HTML are considered as a single class, called the transparent class (*trans*). Most of these files are extracted from a Linux system. All documentations were installed to extract a number of text, HTML and PDF files. A subset of PDF files also came from personal research papers collected over the years. Image files in different formats were extracted from the Linux system as well as downloaded from the Internet. Executable files consist of Linux system commands. MP3 files were downloaded from sites offering free and legal downloads of music.

The compressed files, archives and encrypted files where created by randomly sampling files from the database (MP3 and executable files were excluded from sampling). We performed sampling without replacement; therefore the total number of files used for training and test still remained the same. To obtain encrypted files, we encrypted a file using one of the following ciphers with equal probability: AES-256, CAST-128, Blowfish, Twofish, and Triple DES.

3.2. Settings

Unless otherwise specified, we use nine file types (classes) for our experiments—enc, exe, gz, jpg, mp3, pdf, png, trans and zip. Depending on the size of the byte stream being extracted, the number of files used for each class changes. For example, for the scenario where 64-byte streams are extracted from the beginning of a file, the number of files used for each class in the training phase was 5000. The number of files used for testing varied across class; the minimum is 500 and the maximum was 3700 for this particular scenario.

We use a 3:1 ratio for the number of files used for testing and training (the actual number of files is dependent on the stream-size being extracted). As a rule, we do not consider a file-type if the number of files available is less than 500 for any of the two phases of training and testing; and this will be explicitly specified below, as and when such cases arise.

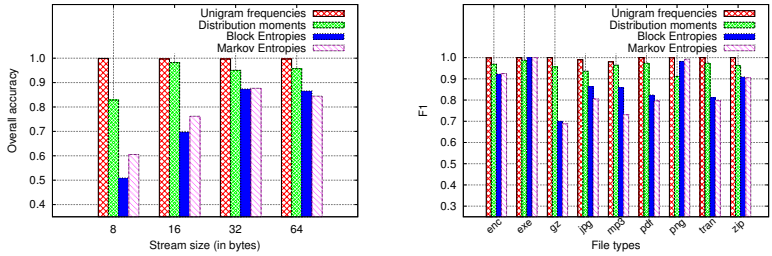
We use Random Forests for classification, as it can efficiently handle large number of features, over a dataset of considerable size. Unless otherwise specified, the four different feature vectors used in the scenarios below are i) Unigram frequencies (Section 2.1), ii) Distribution moments (Section 2.2), iii) Block entropies—feature vector being H_1, H_2, H_3, H_5, as in [3] (Section 2.3), and iv) Markov entropies—feature vector being H_1^M, H_2^M, H_3^M (Section 2.4). Note that the feature H_1 and H_1^M are the same.

3.3. Metrics for evaluation

To evaluate the classifiers using different sets of features, we use two commonly and widely adopted metrics in classification, namely overall accuracy and F_1 score. Overall accuracy, or simply accuracy, is the ratio of the sum of correctly predicted files (across all file types) to the total number of files. For a file type, F_1 score $= 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$, where precision $= \frac{\text{No. of True Positive}}{\#(\text{No. of True Positive + No. of False Positive})}$, and recall $= \frac{\#\text{No. of True Positive}}{\#(\text{No. of True Positive + No. of False Negative})}$.

3.4. Results

We evaluate classifications using the different sets of features, resulting in multiple scenarios. The scenarios and the results are discussed below.



(a) Classification accuracies (b) F_1 for file types

Figure 1. Scenario 1: byte-streams extracted from the beginning of files

Scenario 1 - Feature extraction from the beginning of a file: When a stream of contiguous bytes is extracted from the beginning of a file, naturally the 'fingerprint' (for example, header, magic number, specific strings, etc.) of the file is involved in the phases of classification. Yet, classification using the features extracted from the beginning location is not equivalent to signature matching, which requires an exact match of byte sequences.

Fig. 1(a) plots the accuracies obtained using the different feature vectors, for stream-sizes of 8, 16, 32 and 64 bytes. We observe that feature vector formed of unigram frequencies gives the best results, achieving close to 100% accuracy. While the feature vector of distribution characteristics has a small dimension of four, the classification accuracy is high, and close to the best. Both Block Entropies and Markov Entropies perform relatively worse (in comparison to unigram frequencies and distribution characteristics). The performance obtained with Block entropies is similar to that given in [3] (recall, our work and [3] use the same set of Block entropies as feature vector). However with increasing stream-size, the classification accuracies with these feature vectors are seen to increase.

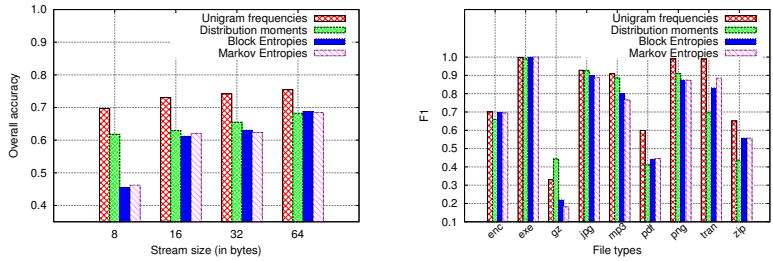
Fig. 1(b) gives the accuracies obtained in classifying each file type, in terms of the F_1 score, when trained and classified using the four different feature vectors. This figure corresponds to the accuracy results for 64 bytes in Fig. 1(a). Entropy features perform worst in classifying gz files.

Scenario 2 - Feature extraction from a fixed location: In this scenario, we skip the very beginning of the file that is highly likely to contain the fingerprints specific to file types. Thus the obfuscation of files by swapping or modifying the header part will not be able to beat the classification system.

Specifically, the feature extraction phase skips the first 100 bytes of a given file, and extracts from the byte location following the 100^{th} byte. Observe that in this scenario, the location of the stream of consecutive bytes is fixed (for both the training and the testing phases).

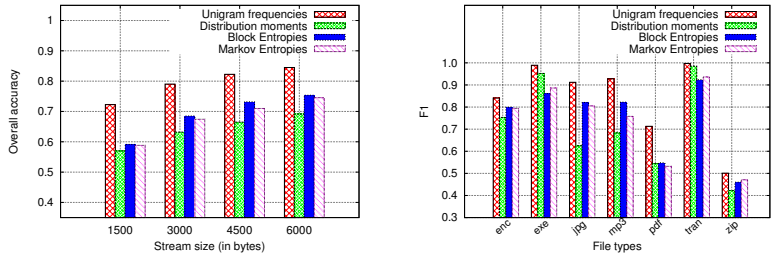
Fig. 2(a) plots the classification accuracies. We observe that the accuracies for all the four feature vectors have come down significantly. The feature vector using unigram frequencies gives the best accuracies, ranging from 73% to 76%. The remaining three feature vectors lead to comparable accuracies. The accuracies also increase with increasing stream size.

For the experiment using streams of 64 bytes, the correspond F_1 score is given in Fig. 2(b). Comparing the classification accuracies of *gz*, *pdf* and *zip* file



(a) Classification accuracies (b) F_1 for file types

Figure 2. Scenario 2: byte-streams extracted after skipping first 100 bytes of a file



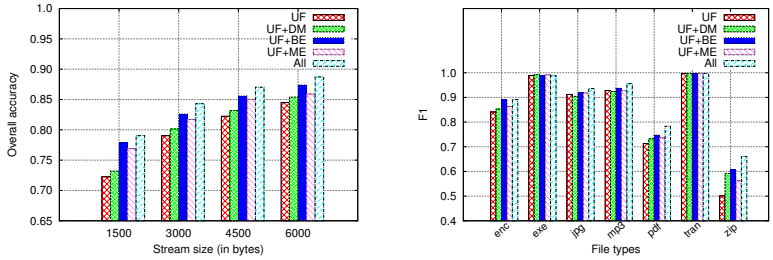
(a) Classification accuracies (b) F_1 for file types

Figure 3. Scenario 3: byte-streams extracted from a random location in a file

types in figures 1(b) and 2(b), it is clear that the header significantly helps in identifying these file types, as header removal brought down the classification accuracy significantly for all feature vectors.

Scenario 3 - Feature extraction from a random location in a file: Here we extract contiguous stream of bytes from a randomly chosen location. To ensure that there is no bias towards the headers of file types, the first 100 bytes are skipped, and a location in a file is selected uniformly randomly. We observed from the experiments that stream sizes in 100s of bytes (from random locations) do not give good accuracy in classifying file types. Therefore, here we provide the accuracies for stream sizes in 1000s of bytes; more specifically, we extract streams of sizes in multiples of a standard data packet size—1500 bytes. Thus the buffer space required to store packets of a single flow (or connection) can be estimated in number of packets.

As the numbers of files for types *gz* and *png* were less than the minimum requirement, we do not consider these file types in this scenario, and the scenarios following. In Fig. 3(a), we plot the accuracies for stream sizes corresponding to one, two, three and four packets. Unigram frequencies still give the best classification accuracy, and the accuracy increases with the number of contiguous bytes used for classification. However, different from the previous scenarios, entropies give better classification results than distribution characteristics in this scenario. We need to keep in mind that the computational cost of entropies is higher than the other features. From Fig. 3(b) (plotted for stream-sizes of 6000 bytes), we ob-



(a) Classification accuracies (b) F_1 for file types

Figure 4. Scenario 4: using combinations of feature vectors

serve that for all feature vectors, pdf and zip file types pose challenge in accurate classification.

Scenario 4 - Combinations of feature vectors: In this section, we experiment combinations of feature vectors. As unigram frequencies have given the highest accuracies in all the previous scenarios, we combine feature vector of unigram frequencies with other feature vectors. UF stands for unigram frequencies, DM for distribution moments, BE for Block entropies, and ME for Markov entropies. We also study the effectiveness of using all feature vectors in classification (denoted by 'All' in the corresponding figures).

Fig. 4(a) gives the accuracies. The highest accuracy, of $\approx 89\%$, is attained for random streams of size 6000 bytes, for the combination of all the four feature vectors ('All' in the figure). In general, combining feature vector of entropies with unigram frequencies is seen to increase the classification accuracy. While adding feature vector of block entropies to the feature vector of unigram frequencies always increases the accuracies of classification of file types, this relative improvement decreases with increasing stream size. Recall that the computational cost of block entropies is a quadratic function of the number of bytes. Computing entropies for larger size streams may potentially slow down the traffic, depending on the capacity of the network and traffic characteristics (number of flows per second, number of packets per second).

Fig. 4(b) gives the F_1 score for stream sizes of 6000 bytes. We observe that, in comparison to unigram frequencies, the combination UB+BE feature vectors increases the accuracy of classification of encrypted files from 84% to 89%. The combination of all feature vectors significantly increases the accuracy of zip file classification from 50% to 66%. The combination also increases the classification accuracy of PDF files from 71% to 78%.

3.5. Discussion

To check if similar performance in terms of accuracy can be achieved when all computations are bounded within quadratic time in the number of bytes extracted, we performed one more experiment by removing the only feature that takes more runtime complexity—H_3^M. This modified set of features, all of which can be computed in linear or quadratic time, is equivalent to the features used in 'All' (UF+DF+BE+ME), but with the third feature in the ME feature vector

removed. The accuracy and F_1 scores obtained were almost the same as those for the 'All' set of features. From this, one can now decide the features (and their combinations) as well as the stream-size to be used for classification depending on the resource capabilities (computational power and buffer sizes) at one's disposal and the network load. One set of features requires only linear time—UF and DF, and another set requires quadratic time—BE and ME (H_1^M, H_2^M). Increasing stream-size for classification results in increased accuracy, but at the cost of both increased buffering and computational time (where, depending on the features, the increase in computational time will be linear or quadratic).

4. Conclusions

In this paper, we conducted studies to determine the right set of features and stream-size for accurately classifying file types in network traffic. Feature vector consisting of unigram frequencies extracted from beginning of files, and which can be computed in $\mathcal{O}(m)$ time, gives the best and close to 100% accuracy. As not more than 64 bytes are required to achieve this performance, the features can be computed in constant time.

As header of files alone cannot be used a reliable source for classification, we also experimented on streams extracted from random locations (and excluding the header part). Our experiments reveal that though unigram frequencies perform good, combining all feature vectors together indeed performs the best. Depending on the resources available at the point of deployment and the network load, (in terms of link capacities, number of flows per second, number of packets per second, etc.), the stream size and feature vectors to be computed may be varied.

Acknowledgment

This material is based on research work supported in part by the Singapore National Research Foundation under NCR Award No. NRF2014NCR-NCR001-034.

References

[1] S. L. Garfinkel, "Carving Contiguous and Fragmented Files with Fast Object Validation," *Digital Investigation*, vol. 4, pp. 2–12, 2007.

[2] N. Beebe, L. Maddox, L. Liu, and M. Sun, "Sceadan: Using Concatenated N-Gram Vectors for Improved File and Data Type Classification," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 9, pp. 1519–1530, Sept 2013.

[3] A. R. Khakpour and A. X. Liu, "An Information-theoretical Approach to High-speed Flow Nature Identification," *IEEE/ACM Trans. Netw.*, vol. 21, no. 4, pp. 1076–1089, 2013.

[4] M. McDaniel and M. Heydari, "Content based file type detection algorithms," in *Proc. of the 36th Annual Hawaii International Conference on System Sciences*, Jan 2003.

[5] W. J. Li, K. Wang, S. Stolfo, and B. Herzog, "Fileprints: Identifying file types by n-gram analysis," in *Workshop on Information Assurance and security (IAW05), United States Military Academy*, 2005, pp. 64–71.