Proceedings of the Twenty-Ninth International
Florida Artificial Intelligence Research Society Conference

Suspiciously Structured Entropy: Wavelet Decomposition of Software Entropy Reveals Symptoms of Malware in the Energy Spectrum

Michael Wojnowicz, Glenn Chisholm, Matt Wolff
Cylance, Inc.
18201 Von Karman Ave.
Irvine, California 92612

Abstract

Sophisticated malware authors can sneak hidden malicious code into portable executable files, and this code can be hard to detect, especially if it is encrypted or compressed. However, when an executable file shifts between native code, encrypted or compressed code, and padding, there are corresponding shifts in the file's representation as an entropy signal. In this paper, we develop a method for automatically quantifying the extent to which the patterned variations in a file's entropy signal makes it "suspicious." A corpus of $n = 39,968$ portable executable files were studied, 50% of which were malicious. Each portable executable file was represented as an entropy stream, where each value in the entropy stream describes the amount of entropy at a particular locations in the file. Wavelet transforms were then applied to this entropy signal in order to extract the amount of entropic energy at multiple scales of code resolution. Based on this entropic energy spectrum, we derive a Suspiciously Structured Entropic Change Score (SSECS), a single scalar feature which quantifies the extent to which a given file's entropic energy spectrum makes the file suspicious as possible malware. We found that, based on SSECS alone, it was possible to predict with 68.7% accuracy whether a file in this corpus was malicious or legitimate (a 18.7% gain over random guessing). Moreover, we found that SSECS contains predictive information not contained in mean entropy alone. Thus, we argue that SSECS could be a useful single feature for machine learning models which attempt to identify malware based on millions of file features.

Introduction

The Entropy Of Malicious Software

A fundamental goal in the information security industry is malware detection. In this paper, we focus our malware detection efforts on the fact that malicious files (e.g. exploits with injected shellcode) commonly contain encrypted or compressed ("packed") segments which conceal malicious code (Brosch and Morgenstern 2006). Thus, the information security industry has been interested in developing methodologies which can automatically detect the presence of encrypted or compressed segments hidden within portable executable files. To this end, entropy analysis has been used, because files with high entropy are relatively likely to have encrypted or compressed sections inside them (Lyda and Hamrock 2007). In general, the entropy of a random variable reflects the amount of uncertainty (or lack of knowledge) about that variable. Chunks of code that have been compressed or encrypted tend to have higher entropy than native code (Lyda and Hamrock 2007).

Suspiciously Structured Entropy

Malicious code, when concealed in a sophisticated manner, may not be detectable through simple entropy statistics, such as mean file entropy. Malware writers sometimes try to conceal hidden encrypted or compressed code; for instance, they may add additional padding (zero entropy chunks), so that the file passes through high entropy filters. However, files with concealed encrypted or compressed segments tend to vacillate markedly between native code, encrypted and compressed segments, and padding, with each segment having distinct and characteristic expected entropy levels. Thus, cybersecurity researchers (Sorokin 2011), have started to pay attention to files with *highly structured entropy*, that is, files whose code flips between various distinguishing levels of entropy through the file.

In order to automatically identify the degree of entropic structure within a piece of software, we represent each portable executable file as an "entropy stream." The entropy stream describes the amount of entropy over a small snippet of code in a certain location of the file. The "amount" of entropic structure can then be quantified, such that we can differentiate, for example, between a low-structured signal with a single local mean and variation around that mean, versus a highly-structured signal whose local mean changes many times over the course of the file.

In this paper, we define *suspiciously structured entropy* as a *particular pattern of entropic structure* which matches those of malicious files. To quantify the suspiciousness of the structured entropy within a piece of software, we develop the notion of a "Suspiciously Structured Entropic Change Score" (SSECS). In this paper, we describe how to calculate SSECS. The derivation of the feature depends upon the notion of a wavelet transform, which we now briefly review.

Brief Overview Of Wavelets

The Wavelet Transform is the primary mathematical operator underlying our quantification of structurally suspicious

entropy. The Wavelet Transform extracts the amount of "detail" exhibited within a signal at various locations over various levels of resolution (Nason 2010). In essence, it transforms a one-dimensional function of "location" (in our case, file location) into a two-dimensional function of "location" and "scale". By using the output of the wavelet transform (the so-called "wavelet coefficients"), it is possible to obtain a series of coarse-to-fine approximations of an original function. These successive approximations allow us to determine the multi-scale structure of the entropy signal, in particular the "energy" available at different levels of resolution.

For this paper, we apply Haar Wavelets, which is a particularly simple family of wavelets whose members are piecewise constant. The Haar Wavelet Transform projects the original entropy signal onto a collection of piecewise constant functions which oscillates as a square wave over bounded support (i.e. assume non-zero values only on certain bounded intervals). Since these piecewise constant functions have supports which vary in their scale (width) and location, the resulting projections describe the "detail" within the signal at various locations and resolutions.

More specifically, the Haar Wavelet Transform is based upon the so called "mother function", $\psi(t)$, defined by:

$$\psi(t) = \begin{cases} 1, & t \in [0, 1/2) \\ -1, & t \in [1/2, 1) \\ 0, & \text{otherwise} \end{cases}$$

a very simple step function. Given the Haar mother function $\psi(t)$, a collection of dyadically scaled and translated wavelet functions $\psi_{j,k}(t)$ are formed by:

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k)$$

where the integers j, k are scaling parameters. The dilation parameter j indexes the level of detail or resolution at a particular stage of analysis, and the translation parameter k selects a certain location within the signal to be analyzed. Note that as the scaling parameter j increases, the function $\psi_{j,k}$ applies to (is non-zero over) successively finer intervals of the signal.

Given a signal $x(t)$ where $t = 1, \ldots, T$, we first rescale the signal so that the final observation occurs at time $t = 1$, and then the so-called "mother wavelet coefficient" at scale j and location k is given by the inner product of the signal with the wavelet. Since we are dealing with discrete signals, the inner product takes the form:

$$d_{j,k} =< x, \psi_{j,k} >= \sum_{t=1}^{T} x(t)\psi_{j,k}(t),$$

One interpretation of this coefficient is that it gives the (scaled) difference between local averages of signal across neighboring chunks or bins. The size of the neighboring chunks is determined by the scaling parameter j.

The family of mother wavelet coefficients, $\{d_{j,k}\}$, enable a "Multi-Resolution Analysis" (MRA) of the signal $x(t)$. In particular, the signal $x(t)$ can be decomposed into a series of approximations $x_j(t)$, whereby each successive approximation $x_{j+1}(t)$ is a more detailed refinement of the previous approximation, $x_j(t)$. The functional approximations are obtained through the wavelet coefficients by the formula:

$$x_{j+1}(t) = x_j(t) + \sum_{k=0}^{2^j-1} d_{j,k}\psi_{j,k}(t)$$

where $x_0(t)$, the coarsest-level functional approximation, is the mean of the full signal. Thus, the collection of mother wavelet coefficients $\{d_{j,k}\}$ store the "details" that allow one to move from a coarser approximation to a finer approximation. Examples of successive functional approximations, in the context of software entropy signals, will be provided in the results section.

Using the wavelet transform, it is possible to "summarize" the overall amount of detail in a signal at various levels of resolution. The total amount of detail at a particular (jth) level of resolution is known as the *energy* at that level of resolution:

$$E_j = \sum_{k=1}^{2^{j-1}} (d_{jk})^2 \tag{1}$$

The distribution of energy across various levels of resolution is known as an *energy spectrum*. Note that the energy at resolution level j is just the squared Euclidean norm of the vector of mother wavelet coefficients from resolution level j. After this step, we have reduced the original signal of size $T = 2^J$ (and resultant wavelet vector of size $T-1$) to a vector of J elements, where each element represents the amount of "energy" at a single level of resolution.

Wavelet-Based Classifiers

The energy spectrum of signals have been very useful features for classifiers such as neural networks. In fact, this combined strategy, whereby the coefficients from a discrete wavelet transform are used as node activations in a neural network, is referred to as a wavelet-neural-network (*WNN*) strategy (see e.g. (Pati and Krishnaprasad 1993)). Using WNN's, researchers have been able to automatically classify lung sounds into categories (crackles, wheezes, striders, squawks, etc.) (Kandaswamy et al. 2004), to automatically determine whether brain EEG scans originated from healthy patients, patients with epilepsy, or patients who were in the middle of having a seizure (Omerhodzic et al. 2013), or to automatically determine whether EMG signals collected from the bicep originated from patients who were healthy, suffering from myopathy, or suffering from neurogenic disease (Subasi, Yilmaz, and Ozcalik 2006).

We refer to the overall strategy of using wavelet coefficients as features in a classifier as a *Wavelet-Based Classifier* strategy. We prefer this term over *WNN*, which, although well-established in the literature, is specific to neural network classifiers. Indeed, in this paper, we choose logistic regression rather than a neural network to model our data, because the logistic regression model provides an "atomic analysis" of the relationship between the wavelet-based features and classification categories.

Suspiciously Structured Entropic Change Score (SSECS)

The initial fundamental problem with applying wavelet-based classifiers to malware analysis is that executable files out in the "wild" have different lengths. This contrasts with controlled observational situations, e.g. those described above, which produce signal samples of fixed length that are held constant across the data set. In controlled observational situations, all samples will produce the same number of features, J, and variation across these set of J features can be immediately associated with a classification variable in a straightforward manner, for example by setting the input layer of the neural network to have J activation notes.

However, in uncontrolled observational contexts, signal lengths can differ wildly from sample to sample. Imagine, for instance, comparing signal A of length 32 (so J=5, and if $E_{f,j}$ represents the energy at resolution level $j = 1, \ldots, J$ for portable executable file f, we would have $E_{a,1}, \ldots, E_{a,5}$) with signal B of length 256 (so J=8, and we have $E_{b,1}, \ldots, E_{b,8}$). How should we compare these two files?

Our solution to this problem is to transform each file's J-dimensional energy spectrum into a single scalar feature, a 1-dimensional "Suspiciously Structured Entropic Change Score" (SSECS). The computation of SSECS is a two-step process: first, we compute the wavelet-based energy spectrum of a file's entropy signal, and second, we compute the file's malware propensity score from that energy spectrum. In our case, we fit a logistic regression model to the binary classification response (malware or not) which uses these wavelet energy features as predictor variables. We fit J separate regression models, one for each file size grouping. Given the Energy Spectrum $\{E_{f,j}\}$, which is the set of wavelet energies for each resolution level $j = 1, \ldots, J$ of portable executable file f, the logistic regression model estimates $\widehat{P}_f$, the predicted probability that file f is malware, by the formula

$$\widehat{P}_f = \frac{1}{1 + exp[-E_{f,j} \cdot \beta_j^{(J)}]}$$

where $\beta_j^{(J)}$ is a model parameter, known as a "logistic regression coefficient", from the Jth logistic regression model. This number, $\widehat{P}_f$ is what we refer to as the SSECS.

Data

Data were a set of n=39,968 portable executable files from a Cylance repository. 19,988 (50.01%) of these files were known to be malicious, and the remaining files were benign.

Method

Constructing the entropy stream

To compute the entropy of an executable file, the original file, represented in hexadecimal (00h-FFh), is split into non-overlapping chunks of fixed length, typically 256 bytes. For each chunk of code, the entropy is then computed using the formula below:

$$H(c) = -\sum_{i=1}^{m} p_i(c) \log_2 p_i(c), \quad (2)$$

where c represents a particular chunk of code, m represents the number of possible characters (here, n=256), and p_i is the probability (observed frequency) of each character in the given chunk of code. The entropy for any given chunk then ranges from a minimum of 0 to a maximum of 8.

Computing the Suspiciously Structured Entropic Change Score (SSECS)

The procedure for computing the suspiciously structured entropic change score (SSECS) is as follows:

1) Partition data set by size: Group sampled files into $j = \{1, \ldots, J\}$ groups, where $j = \lfloor log_2 T \rfloor$ and T is the length of the file's entropy stream:

2) Iterate: For all files which fall into the jth length group

2a) Compute Haar Discrete Wavelet Coefficients: The discrete wavelet transform takes as input a discrete series of size $T = 2^J$ observations. Because the transform requires the series to have a dyadic length, if the number of observations in the executable file's entropy stream is not an integer power of 2, we right-truncate the series at value $2^{\lfloor log_2 T \rfloor}$. The so called "mother" wavelet coefficients, d_{jk}, describe the "detail" at successively fine-grained resolutions. In particular, the mother wavelet coefficients are indexed such that $j \in \{1, \ldots, J\}$ represents the resolution level, ordered from coarse-grained to fine-grained, and $k \in \{1, \ldots, K = 2^{j-1}\}$ represents the particular location (or bin) of the entropy signal at that resolution level. At each resolution level j, the signal is divided into $N_j = 2^{j-1}$ non-overlapping, adjacent bins such that each bin includes $B_j = 2^{J-j}$ observations. Note that the number of bins, K, increases as j increases to finer resolutions. The mother wavelet coefficient at index (k, j) is then given by:

$$d_{kj} = \frac{1}{s_j}\left(\sum_{i=(2k-1)Bj+1}^{2kBj} y_i - \sum_{i=(2k-2)Bj+1}^{(2k-1)Bj} y_i \right) \quad (3)$$

where the scaling factor is $s_j = (\sqrt{2})^{J-j+1}$ and is necessary for the wavelet transform to preserve the size (norm) of the signal. There are T-1 mother wavelet coefficients.

2b) Compute Wavelet Energy Spectrum: The *wavelet energy spectrum* summarizes the "detail" or "variation" available at various resolution levels. The energy spectrum is computed as a function of the mother wavelet coefficients, d_{jk}. In particular, the "energy", E_j, of the entropy stream at the jth resolution level is defined by Equation 1.

2c) Compute Wavelet Energy Suspiciousness Now we use the wavelet energy spectrum to determine the "propensity" of each file to be malware (i.e. its suspiciousness). Computing this propensity requires training. We use 5-fold validation.

2c1) Partition The Current Sample Of Files: Split the entire set of F_J files which are of the appropriate size into 5 mutually exclusive subsets $F_J^1, \ldots, F_J^5$, each of which represents exactly 20% of the entire sample.

2c2) Iterate: For each subset F_J^i, where $i \in \{1, \ldots, 5\}$

2c2a) Fit a logistic regression : Fit a logistic regression model on the other four subsets $\{F_J^k : k \neq i\}$ which fits the class variable (malware or not) as a function of the wavelet energy spectrum. The logistic regression model will produce a set of beta coefficients to weight the strength of each resolution energy on the probability of being malware.

2c2b) Calculate malware propensity: Use the logistic regression model above to then make a prediction about files in subset F_J^i. In particular, use the model learned in step 1c2a to calculate the predicted probability that each file in set F_J^i is malware, given its wavelet energy spectrum. This malware propensity (i.e., predicted malware probability) lies within the interval $[0, 1]$, and is what we call the Suspiciously Structured Entropic Change Score (SSECS).

Results

The Logistic Regression Model for One File Size Group, $J = 5$

How does the model transform these wavelet energy spectra into predictions about whether the file is malware (that is, into a Suspiciously Structured Entropic Change Score)? To illustrate, we consider the subset of n=1,599 files in our corpus belonging to file size group $J = 5$. Because these files can be analyzed at $J = 5$ different resolutions, we extract 5 features from each file, with each feature representing the energy at one level of resolution in the file's entropy stream.

For illustrative purposes, we begin by analyzing the wavelet energy spectrum for two files from this size category, as they embody more general trends in the energy patterns of malicious versus clean files. Figure 1 shows wavelet-based functional approximations for two different entropy streams. The left column of the plot depicts the entropy signal from File A, which is legitimate software, whereas the right column of the plot depicts the entropy signal from File B, which is malware. Reading these columns from top to bottom, we see that the wavelet transform produces successively detailed functional approximations to these files' entropy signals. The title above each subplot shows the wavelet energy, as computed in Equation (1) in the text, of the signal at a particular resolution level. The wavelet energy is simply the sum of the squares of the scaled differences in the mean entropy levels, where the differences are only taken between even/odd index pairings (i.e. the algorithm takes the differences $mean_{bin2} - mean_{bin1}, mean_{bin4} - mean_{bin3}$, and so forth). Thus, we can gain some visual intuition about how the energy spectra can be derived from these successive functional approximations.

Based on this entropic energy spectrum decomposition (or distribution of energy across various levels of resolution), the model believes that File A is legitimate software, whereas File B is malware. Investigating this conclusion, we

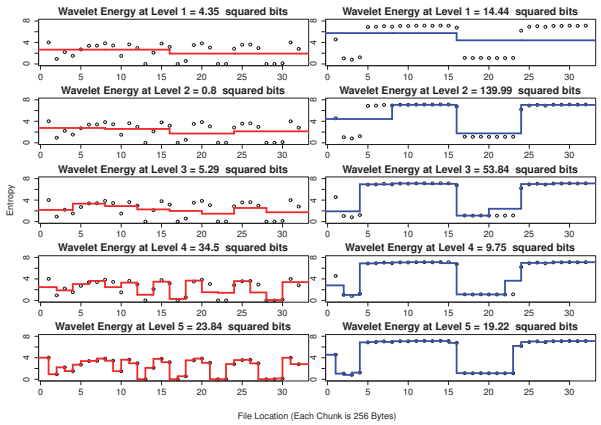

Figure 1: Wavelet-based functional approximations, and the corresponding wavelet energy spectrum, for the entropy signals of two representative portable executable files from one file size group.

see that these two files have radically different wavelet energy distributions across the 5 resolution levels. The legitimate software (File A) has its "entropic energy" mostly concentrated at finer levels of resolution, whereas the piece of malware (File B) has its "entropic energy" mostly concentrated at coarser levels of resolution. For the clean file, the energy in the entropy stream is concentrated at the resolution levels $j = 4$ and $j = 5$ (where the energy is 34.5 and 23.84 squared bits, respectively). For the dirty file, the energy in the entropy signal is concentrated at coarser levels of analysis, peaking especially strongly at level $j = 2$ (where the energy is 139.99 squared bits).

The fit of the logistic regression model (for both raw and normalized features) is summarized in Table 1. Note that for the entire table, numbers outside the parentheses represent results for the normalized features, whereas numbers inside the parentheses represent results for raw features. The two "Energy" columns list the energy at all five levels of resolution for these two files. The "Value of β_j" column describes the estimated beta weight in a logistic regression fitting file maliciousness to the five wavelet energy values, based on a corpus of n=1,599 files. The "P-value" column describes the probability of getting the test statistic we observed (not shown, it is a function of the data) under the hypothesis that there is no relationship between energy at that level and file maliciousness. The codes are: $* = p < .05, ** = p < .01, *** = p < .001, **** = p < .0001, ***** = p < .00001$. The "Malware Sensitivity" represents the estimated change in the odds that a file is malware associated with an increase of one unit in the corresponding feature. It is calculated by $(e^{\beta} - 1) \times 100\%$. For the normalized values (those outside the parenthesis), an increase of one unit refers to an increase of one standard deviation.

Based on these logistic regression beta weight (β_j) values, we see that the two sample files from Figure 1 are indeed representative of a larger trend: having high energy at resolu-

tion levels 1,2 and 3 (the coarser levels) is associated with a higher probability of the file being malware (since those β_j's are positive), whereas having high energy at levels 4 and 5 (the finer levels) is associated with a lower probability of the file being malicious (since those β_j's are negative). Moreover, these associations appears to be reflective of trends in the larger population of files, since the p-values are largely strongly statistically significant. This finding makes sense if artificial encryption and compression tactics tend to elevate moderate to large sized chunks of malicious files into "high" entropy states.

Logistic Regression Models For All File Size Groups

Do the trends found in the single level analysis of $n = 1,599$ files hold up in the full corpus of $n = 39,968$ files? In particular, regardless of file size, can we corroborate the simply stated conclusion that "malware tends to concentrate entropic energy at relatively coarse levels of resolution?" And if so, where is the dividing line between "coarse" and "fine"?

In Figure 2, we summarize the results of logistic regression models fit to each file size grouping separately. The plot shows logistic regression beta coefficients for determining the probability that a portable executable file is malware based upon the magnitude of file's entropic energy at various levels of resolution within the code. Positive betas (red colors) mean that higher "entropic energy" at that resolution level is associated with a greater probability of being malware. Negative betas (blue colors) mean that higher "entropic energy" at that resolution level is associated with a lower probability of being malware. For both colors, stronger intensities represent stronger magnitudes of the relationship between entropic energy and malware. Mathematically, the dot product between a file's energy spectrum and these beta weights determine the fitted probability that the file is malicious. Thus, the *Danger Map* interpretation arises as follows: For any file size grouping (or row), files that have high energies in the red spots and low energies in the blue spots are significantly more likely to be "dangerous." Conversely, files that have low energies in the red spots and high energies in the red spots are significantly more likely to be "safe."

Taking this *Danger Map* into consideration, we draw the following conclusions:

- The full analysis supports the "coarse-energy-is-bad, fine-energy-is-good" mantra to a first approximation. Visually, most diagonal elements of the matrix are blue (and also more blue than the off-diagonals). Thus, across most file sizes, high energies at the finest-level of resolution appear to be indicative of file legitimacy, and high energies at coarse levels of resolution are often associated with suspiciousness.
- However, this mantra is too simplistic to be satisfying as a full description. What qualifies as a suspicious pattern in the wavelet decomposition of a file's entropy stream appears to be much more complex. For example, the appearance of the double diagonal bands in blue suggest somewhat regular vacillations in terms of how

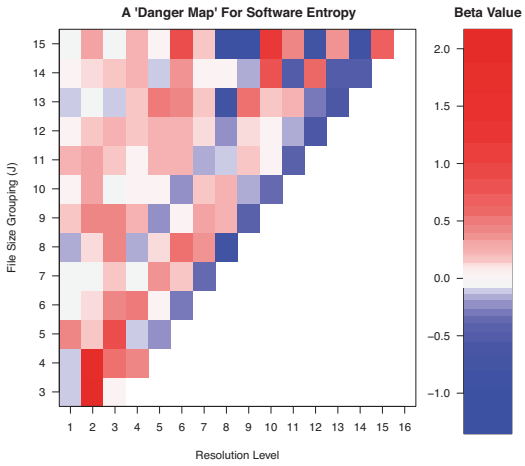

Figure 2: A *Danger Map* for entropy patterns within a piece of software.

"suspicious" high entropic energy would look at a particular level of resolution. Indeed, the particular patterning depicted in the Danger Map provides a statistically significantly better description of malware than random (baseline-informed) guessing alone. Likelihood ratio tests comparing the fit of the size-specific models (where the beta coefficients of each size-specific model are given by the specific colorings in the corresponding row of the Danger Map) versus the fit of models with no features (interpretable as a uniform color across rows, where the intensity of the color is determined by baseline malware rates, independent of the wavelet energy spectrum) yield the test statistics below. Moving from bottom (J=3) to top (J=16) of the figure, we have

$$\chi^2(3) = 198.36, \quad \chi^2(4) = 563.51, \quad \chi^2(5) = 257.52,$$
$$\chi^2(6) = 235.09, \quad \chi^2(7) = 150.11, \quad \chi^2(8) = 585.57,$$
$$\chi^2(9) = 662.22, \quad \chi^2(10) = 283.24, \quad \chi^2(11) = 385.33,$$
$$\chi^2(12) = 305.04, \quad \chi^2(13) = 233.39, \quad \chi^2(14) = 116.17,$$
$$\chi^2(15) = 61.88, \quad \chi^2(16) = 31.44$$

All of these are statistically significant at the $\alpha = .05$ level.

The Predictive Power of SSECS

How can we use the information distributed across the "Danger Map" to construct a single number which could score a piece of software's suspiciousness based on the wavelet decomposition of its entropy signal? We studied the predictive performance of SSECS in identifying malware by constructing a hold-out test set of $n = 7,991$ files and found:

1. SSECS as a single feature improved predictions of malware, within a balanced sample of malware and legitimate software, from 50% to 68.7% accuracy. This makes SSECS a particularly impressive feature, considering that

Resolution			*Energy Spectra*		*Statistical Model For File Size $J = 5$*		
Level	*# Bins*	*Bin Size*	*File A*	*File B*	*Value of β_j*	*$P-value$*	*Malware Sensitivity*
1	2	16	-0.39 (4.35)	-0.01 (14.44)	0.448 (0.017)	*****	+56.5% (+1.7%)
2	4	8	-0.79 (0.80)	6.27 (139.99)	0.174 (0.008)	*	+19.0% (+0.89%)
3	8	4	-0.48 (5.29)	2.18 (53.83)	0.847 (0.046)	*****	+133.2% (+4.74%)
4	16	2	1.42 (34.50)	-0.37 (9.75)	-0.106 (-0.008)	n.s.	-10.0% (-0.75%)
5	32	1	1.77 (23.84)	1.19 (19.22)	-0.240 (-0.030)	**	-21.4% (-2.99%)

Table 1: Investigating the relationship between the entropic Wavelet Energy Spectrum and maliciousness for files of size J=5.

most machine learning models of malware consist of millions of features.

2. SSECS provides predictive information beyond what is contained in a mean entropy feature. A model with mean entropy as a single feature achieved 66.2% predictive accuracy. Thus, mean entropy is indeed also a very impressive single predictor of malware (perhaps not surprisingly given its prevalence in the literature). However, unlike mean entropy, the wavelet energy spectrum detects suspicious patterns of entropic *change* across the code of the executable file. We found that a 2-feature model which includes both mean entropy and SSECS achieves 73.3% predictive accuracy (so adding wavelet-based information to the model yields a 7.1% boost in predictive accuracy beyond what is obtained by mean entropy alone).

3. SSECS provides predictive information beyond what is contained in a "standard deviation of entropy" feature. A skeptic might ask: why not simply use standard deviation, a more commonly used and more computationally straightforward measure of variation? Standard deviation is useful, but a relatively cruder measure of variation, as it operates on only a single spatial scale. Indeed, a 2-feature model which includes both mean entropy and standard deviation achieves merely 70.4% predictive accuracy.

4. Overall, SSECS is a valuable contributor to a collection of entropy-based features for predictive modeling of malware. We construct a somewhat more comprehensive model of *Entropic Suspiciousness*, which includes both SSECS as well as 8 simple statistical summary features of the entropy signal which may be relevant for malware detection: mean, standard deviation, signal-to-noise ratio, maximum entropy, percentage of the signal with "high" entropy ($\geq$ 6.5 bits), percentage of the signal with zero entropy, and length and squared length of the signal. This model made correct predictions 74.3% of the time.

Discussion

All together, we have found that the Suspiciously Structured Entropic Change Score (SSECS) is a single feature which would appear to be useful in machine learning models of malware detection. Malware authors have been known to deliberately tinker with software to incorporate malicious code, but then to conceal this code by encryption, compression, and padding. By design, SSECS helps to capture the degree to which a portable executable file exhibits suspicious patterns of shifting entropy within the code of portable executable files. Moreover, SSECS (and a broader notion of entropic suspiciousness that goes beyond mean entropy) could easily be combined with other classes of data mining features, such as n-gram features (Kolter and Maloof 2004) and strings (Schultz et al. 2001), to construct more powerful automatic malware classifiers.

References

Brosch, T., and Morgenstern, M. 2006. Runtime packers: The hidden problem. *Black Hat USA*.

Kandaswamy, A.; Kumar, C. S.; Ramanathan, R. P.; Jayaraman, S.; and Malmurugan, N. 2004. Neural classification of lung sounds using wavelet coefficients. *Computers in Biology and Medicine* 34(6):523–537.

Kolter, J. Z., and Maloof, M. A. 2004. Learning to detect malicious executables in the wild. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 470–478. ACM.

Lyda, R., and Hamrock, J. 2007. Using entropy analysis to find encrypted and packed malware. *IEEE Security & Privacy* (2):40–45.

Nason, G. 2010. *Wavelet methods in statistics with R*. Springer Science & Business Media.

Omerhodzic, I.; Avdakovic, S.; Nuhanovic, A.; and Dizdarevic, K. 2013. Energy distribution of eeg signals: Eeg signal wavelet-neural network classifier. *arXiv preprint arXiv:1307.7897*.

Pati, Y. C., and Krishnaprasad, P. S. 1993. Analysis and synthesis of feedforward neural networks using discrete affine wavelet transformations. *Neural Networks, IEEE Transactions on* 4(1):73–85.

Schultz, M. G.; Eskin, E.; Zadok, E.; and Stolfo, S. J. 2001. Data mining methods for detection of new malicious executables. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, 38–49. IEEE.

Sorokin, I. 2011. Comparing files using structural entropy. *Journal in computer virology* 7(4):259–265.

Subasi, A.; Yilmaz, M.; and Ozcalik, H. R. 2006. Classification of emg signals using wavelet neural network. *Journal of neuroscience methods* 156(1):360–367.