Structural Feature Engineering approach for detecting polymorphic malware

Emmanuel Masabo[1]
Dept. of Networks
Makerere University
Kampala, Uganda,
Email:
masabem@gmail.com

Kyanda Swaib Kaawaase[2]
Dept. of Networks
Makerere University
Kampala, Uganda
Email:
kswaibk@cis.mak.ac.ug

Julianne Sansa-Otim[2]
Dept. of Networks
Makerere University
Kampala Uganda
Email: sansa@cit.ac.ug

Damien Hanyurwimfura[2]
Dept. of Comp. & Soft. Eng.
University of Rwanda
Email: hadamfr@gmail.com

***Abstract*—Currently, malware are distributed in a polymorphic form. There are very smart and obfuscated. This serves the purpose of hardening detection or simply making it impossible. Researchers have mainly resorted to static analysis, dynamic analysis or a combination of both in attempting to find advanced solutions to polymorphic malware detection problems. This paper presents a novel simple feature engineering approach in terms of extracting, analyzing and processing static based features for efficient detection of polymorphic malware. K-NN algorithm is used to build the detection model. Our experiments achieve a detection accuracy of 98.7% with 0.014% False Positive Rate (FPR) on a relatively small dataset.**

Keywords— Polymorphic malware, static analysis, machine learning

I. INTRODUCTION

There is a high growth of malicious programs created every day[1]. These threats exploit vulnerabilities in computers, applications and cyber physical devices [2], [3]. These threats hinder security in terms of privacy and data integrity [4], [5]. For example, recently, a malware called WannaCry [6] exploited vulnerability holes in Microsoft operating systems and attacked many machines around the globe by encrypting data and therefore demanding ransom payments for restoring data. Most of these threats are spread directly from Internet and are able to communicate with the outside world and can remotely receive commands from the owners [7]. The reasons behind these actions are, but not limited to steal information, destroy user data or take custody of people's information by encrypting it and make it inaccessible till a ransom is paid by the victim and the data is decrypted [7]. The anti-virus industry and security researchers constantly try to develop solutions that attempt to address this global malware security threats. However, there are still many challenges in terms detecting polymorphic malware due to their smartness in morphing themselves in unlimited number of copies, thus confusing and defeating the detection systems[8].

The success of every detection approach depends on the analysis and features used in the construction of the detection model[9]. Features are the characteristics that identify the malware[10]. Therefore, it is necessary to have good indicative features[11] well engineered to yield better results. In order to extract features during analysis, researchers mainly use two techniques which are: static analysis[12] and dynamic analysis[13]. With static analysis, malware are analyzed without being executed and structural features are extracted [14], [15]. This technique reduces the risks that might be caused by security breaches, but has also some limitations regarding detecting live malware functionalities[10]. On the other side, with dynamic analysis, malware are executed, monitored and behavioral features are extracted for further analysis [10].

This research mainly focuses on engineering statically extracted features in terms of preprocessing, selection and overall detection model construction. The main contribution of this research is to build an easy, fast and efficient structural based polymorphic malware detection approach. Our contribution embodies the following three things:

1. We provide an easy preliminary analysis approach towards the development of a stronger polymorphic malware detection framework.
2. We produce an experimental mechanism that proves the above assumption.
3. We come out with promising detection accuracy and very few false positive.

The rest of the paper is organized as follows. Section 2 describes the related work on malware static analysis, techniques used to build detection models and the results obtained. Proposed approach is introduced in section 3. Section 4 gives details on implementation and discussion. Finally, conclusion and future work suggestions are highlighted in section 5.

II. RELATED WORK

Static analysis[16] is a process whereby information about malware can be extracted without being executed. Static analysis is therefore safe due to the fact that it is not necessary to execute malicious code during analysis[17]. A basic information about the malware can be revealed such as file

version, file format and file size. However, a deeper code inspection[18] in terms of code/structure analysis and more dissection of malicious program reveals more about malware specific functionalities. This requires the knowledge of assembly language, compiler code and operating system concepts [17].

Various static analysis approaches have been developed in attempt to detect malware and their variants. Horng et al.[19] presented a static malware detection system that uses data mining methods for malware detection. Features extracted included PE header, API functions inside DLLs. Feature subset has been calculated using information gain and principal component analysis. The detection accuracy has been 99.6% as given by J48 classifier. Islam et al.[20] presented a method that extracts static features of function lengths and printable strings. K-fold cross validation was used and the classification accuracy has been 98%.

Venkatesh et al. [17] proposed a detection approach that extracts hexadecimal dumps and constructs n-grams that are used as features for classification. They used a dataset of 100 samples, of which 90 were benign and 10 were of a spyware family. Common Feature based extraction (CFBE) and frequency based feature extraction were used to extract features as well as generating a reduced feature set. Naive Bayes was used for classification. The detection rate was 72.7%. Pasha et al.[21] developed a system called "Malwise" that uses static control graph features. They also created unpacking mechanism that was efficient and fast in unpacking files packed by known commonly tools.

Malwise system was able to detect malware and their variants. It was tested on 15000 real malware and could calculate up to 88% probability that a new malware detected is the variant of an existing one. Cesare et al.[22] implemented a flow-based malware variant detection system. They extracted a set of control flow graphs in malware and decomposed the set of graphs into smaller fixed size k-subgraphs. The minimum matching distance between decompiled flow graphs was computed. Their system was able to detect malware variants with limited false positives. A comparison based on features and detection accuracy of different approaches is given in table1.

TABLE 1: IDENTIFYING GAPS IN PREVIOUS WORKS

Reference	Features	Detection accuracy	Gap
[19]	PE header, API functions	99.6%	Considering more features could improve the detection
[20]	function lengths, printable strings	98%	Considering more features could improve the detection
[17]	hexadecimal dumps	72.7%	Very low detection rate
[21]	control graph	88%	Low detection rate

III. PROPOSED APPROACH

The proposed approach is shown in figure 1 and consists of the following steps: dataset collection, feature extraction, feature transformation, feature selection and classification.

A. Dataset collection

The data sample used in this research consists of 10 variants of potao polymorphic malware downloaded from github malware zoo [23], 59 other mixed malware downloaded from nothink [24] and 10 benign files collected from fresh windows operating system installation.

B. Feature extraction

During analysis, structural features have been extracted using tools like VirusTotal, IDA pro, PE Explorer and DetectItEasy. IDA pro and PE explorer have been used to disassemble and extract operation codes (Opcodes) at the entry point of each file used in experiments.

IDA pro is a disassembler software used to reverse engineer binaries by statically revealing their functionalities. IDA pro is the choice of many malware analysts. It supports Portable Executable (EXE) file formats.

PE explorer is another tool used in static analysis. It provides facilities to navigate through the parts of the executable file, view import and export functions, view header information and perform basic disassembly. It has been used in this research to find file sizes, binary digital signatures and opcodes.

DetectItEasy (DIE) is a tool that can detect file types, packer type and section information. It has been used to get feature information related to sections and file packing.

VirusTotal (VT) is a popular online service used to analyze different file formats and URLs. It relies on connected antivirus engines, which identify the malicious file and give basic report once a file is found to be malicious. VT has been used to get feature information about packing, uninitialized data size and packing.

Feature extraction process is described in Algorithm 1. Extracted features are as follows:

- *PackingStatus*—most polymorphic malware are encrypted and compressed. This feature reveals this obfuscation characteristic.
- *NumberOfSections*—obfuscated malware have fewer sections than those of non-obfuscated files because some sections are hidden. This feature provides the number of accessible sections.
- *Entropy*—for obfuscated malware, entropy value is high, which means a high level of disorder in the file content. This feature provides the computed entropy value.
- *InitializedDataSize*—size of initialized data section in bytes.
- *CodeSize*— the code size is smaller for most obfuscated malware compared to non-obfuscated files.
- *EntryPoint*—starting address of an executable file.

- *DigitaSignatureStatus*—most malware file are not genuine. Their digital signature status can be revealed by this feature.
- *OpCodeAtEntryPoint*—this feature gives the opcode of malware entry point. This enables the malware to execute the malicious code before giving control to the original entry point (OEP).
- *Packer type* and *file extensions* have also been used as features.

For packed malicious files, the original entry point (OEP) was PUSHAD, whereby for non-packed files, the original entry point was PUSH, MOV or CMP. Packing is a technique used by malware developers to obstruct detection and analysis. With analysis, the file size is reduced, the code is encrypted and disorder known as entropy is introduced into the file content. For example, a packed file will have fewer strings and sections than the unpacked one and its entropy is high. Most of the information is not visible. Packed samples must be unpacked for a successful static analysis. We found that among the packed files, UPX was the mostly used packer. Other files were packed using Armadillo packer. A packed program is characterized by the highest level of entropy. Entropy is a measure of disorder or randomness in a file[25]. Packing can also be used by genuine software vendors to protect their applications or protect against license cracking. Shannon Entropy algorithm [25] discussed below is used for entropy calculation:

$$H(X) = -\sum_{i=1}^{n} p(i) \log_b P(i) \quad (1)$$

where $H(X)$ is the measured value of the entropy, $P(i)$ is the probability of ith unit of information in the series of n variables for the event X.

Entropy has a measurement scale of 0 to 8. The closer to 0, the more orderly the data is. The highest entropy value >= 7 shows a high level of disorder in the file. Figure 2 shows entropy distribution among benign and malware files in the provided dataset as calculated using equation 1. The malware samples have the highest entropy when compared to benign.

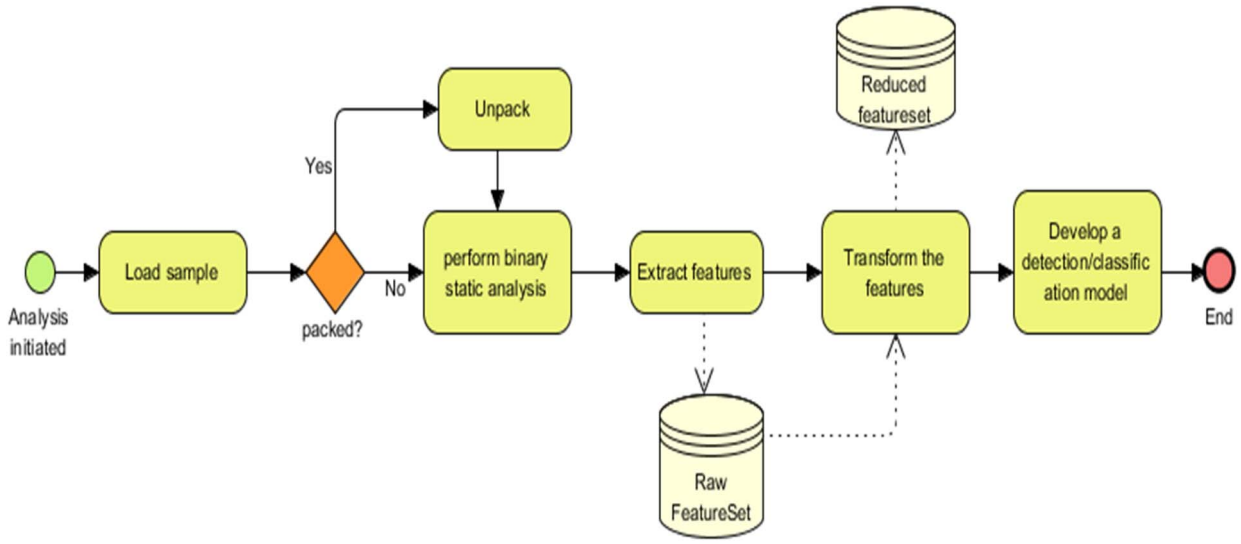

Fig 1: Static based polymorphic malware detection process

Algorithm1: Feature_extractor(S_{mb})

Input: $S_{mb} = S_m \cup S_b$ //set of polymorphic malware //and benign files
Output: F_{raw} // Extracted raw features
$F_{raw} = \emptyset$
FOREACH $s \in S_{mb}$ **DO**
IF $i \in S_m$ and i is packed **THEN**
// S_m is a subset of S_{mb} containing //polymorphic malware

Unpack;
f=Extract(PackingStatus,NumberOfSections,Entropy,InitializedDataSize,CodeSize,EntryPoint,DigitaSignatureStatus,OpCodeAtEntryPoint);
$F_{raw} \leftarrow F_{raw} + f$
END
IF $i \in S_b$ **THEN**
// S_b is a subset of S_{mb} containing //benign files
f=Extract(PackingStatus,NumberOfSections,Entropy,InitializedDataSize,CodeSize,EntryPoint,DigitaSignatureStatus,OpCodeAtEntryPoint);
$F_{raw} \leftarrow F_{raw} + f$
END
END

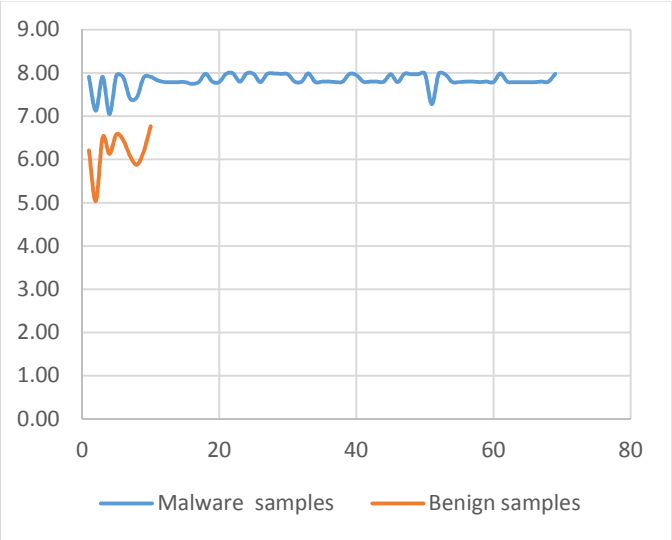

Fig 2: Entropy distribution among samples.

C. Feature selection

Impurities are removed from the original feature set where misleading information is removed. The feature set dimensionality is reduced by only retaining more reliable key features and rejecting the ones that could degrade the classification performance. Algorithm 2 shows the proposed feature reduction process.

Algorithm2: Feature_selector(F_{raw})

Input: F_{raw} //set of raw extracted features
Output: F_{select} // selected relevant features
$F_{select} = \emptyset$
FOREACH $x_i \in F_{raw}$ **DO**
Calculate x_i class predictive ability
Evaluate inter-correlation among x_j, where $j \leq i$
// retain features with high class //predictive ability with low //inter-correlation
x_k = most relevant (x_j),
where $k \leq j$;
Add $F_{select} \leftarrow F_{select} + x_k$
END

Class predictive ability inter-correlation are implemented based on Correlation based feature selection(CFS) algorithm[1] discussed below:

$$Merit_{S_k} = \frac{k\overline{r_{cf}}}{\sqrt{k+k(k-1)\overline{r_{ff}}}} \quad (2)$$

where $\overline{r_{cf}}$ is the average value of feature classification correlations, and $\overline{r_{ff}}$ is the average value of feature-feature correlations.

CFS will finally be computed according to equation 2

$$CFS = \max_{S_k}\left[\frac{r_{cf_1}+r_{cf_2}+\cdots+r_{cf_k}}{\sqrt{k+2\left(r_{f_1f_2}+\cdots+r_{f_if_j}+\cdots+r_{f_kf_1}\right)}}\right] \quad (3)$$

, where r_{cf_i} and $r_{f_if_j}$ variables are correlations.

D. Developing a detection model

For developing a detection model, a machine learning technique called K-Nearest Neighbor has been used. The principal behind this algorithm is the computation of a similarity measure among samples and based on highest similarity threshold, the most identical samples are identified [1]. KNN is discussed below:

1. Let $x_i^{(J)}$ represents all training examples, where i is the number of features and j is the number of instances.
2. Let k be the number of nearest neighbors determined beforehand in building (K-NN) model,
3. Any distance between the targeted instance ($x_i^{(b)}$) and all training examples $x_i^{(j)}$. Euclidean distance will be computed as:

$$dist\left(x_i^{(b)}, x_i^{(j)}\right) = \sqrt{\sum_{a=1}^{i}(x_a^{(b)} - x_a^{(j)})^2}, \text{where } a \leq i \text{ and } b \leq j \quad (4)$$

4. Identify all categories of training instances for the sorted values under k.

IV. Implementation and Discussion

A. Implementation

Implementation was done using a controlled virtual environment. Ubuntu Linux Operating System was to host a virtual Windows 7 guest machine. Oracle Virtual Box hypervisor was used to contain the virtual machine. Analysis tools discussed in section 3 were installed on the guest machine. Feature extraction process has been done as explained by algorithm 1 and feature selection has been done using algorithm 2. KNN algorithm has been used to develop a detection model. Weka tool was used in designing and evaluating the detection model. Cross validation with 10 folds has been used to evaluate our model. The approach was able to achieve a performance of 98.7% detection accuracy. In table 2 and figure 3, we show the detection accuracy of the proposed approach. We wish to mention that the dataset used in this work is not exactly the same like the ones in the works discussed in section 2. Thus, we don't provide a comparison performance between this work and the previous ones. However, the subsequent works will consider this option to evaluate the strengths and weaknesses of the proposed approach. The focus of current research is on how to provide a simple feature engineering mechanism that can lead to satisfying results in terms of detecting polymorphic malware.

TABLE 2: DETECTION ACCURACY

	Sample size	Detected	TP
Polymorphic instances	10	10	1
Other malware	59	58	0.983
Benign	10	10	1

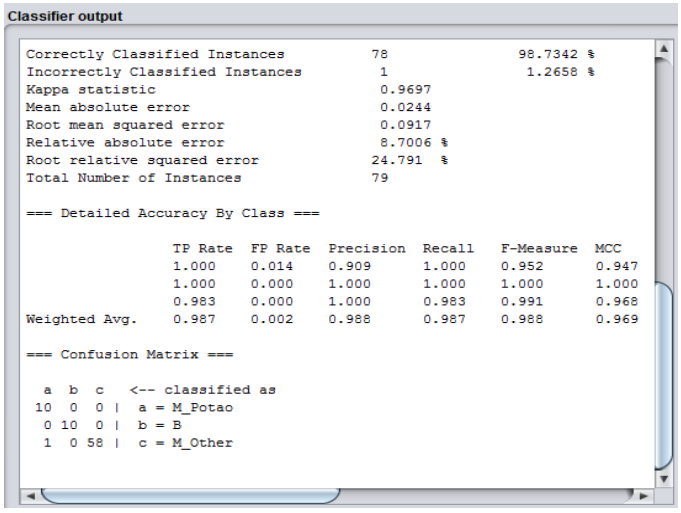

Fig 3: Detection of polymorphic malware

B. Discussion

The proposed approach performs well for a small chosen dataset as per results presented in this section. This remains to be evaluated on very large sample to assess its efficiency. Some samples are obfuscated with packing and encryption techniques. This hardens the analysis task as static analysis is only successful when files are fully unpacked and the original entry point is revealed. In case of failure to unpack some samples due to unavailability of appropriate tool, the sample is kept suspicious and entropy is strongly used as an influencing classification feature.

V. Conclusion

Polymorphic malware can be identified by well-engineered structural features obtained through static analysis. In this paper, we proposed a feature engineering approach that consists of choosing the best features that represent the file statically and transform them using suitable algorithms according to the nature of the problem. The given approach has produced a detection accuracy of 98.7% with 0.014% False Positive (FP). As an ongoing work, this idea will be evaluated on bigger dataset and different classifiers will be utilized to assess the performance. We also plan to incorporate more features such as virtualization awareness, time logic bomb, etc... We will as well incorporate behavioral features obtained through dynamic analysis to develop a more crosscutting based detection approach.

References

[1] E. Masabo, J. Sansa-otim, and D. Hanyurwimfura, "Integrated Feature Extraction Approach Towards Detection of Polymorphic Malware In Executable Files," *International Journal of Computer Science and Security (IJCSS*, vol. 11, no. 2, pp. 25–33, 2016.

[2] Z. A. Bhuiyan, T. Wang, T. Hayajneh, and G. M. Weiss, "Maintaining the Balance between Privacy and Data Integrity in Internet of Things," in *Proceedings of the 2017 International Conference on Management Engineering, Software Engineering and Service Sciences*, 2017.

[3] Z. A. Bhuiyan, M. Zaman, G. Wang, T. Wang, and J. Wu, "Privacy-Protected Data Collection in Wireless Medical Sensor Networks," in *Networking, Architecture, and Storage (NAS), 2017 International Conference*, 2017, pp. 1–2.

[4] Z. A. Bhuiyan and J. Wu, "Trustworthy and protected data collection for event detection using networked sensing systems," in *Sarnoff Symposium, 2016 IEEE 37th*, 2016.

[5] M. Z. A. Bhuiyan, T. Wang, and G. Wang, "Event Detection through Differential Pattern Mining in Cyber-Physical Systems," *IEEE Transactions on Big Data*, 2017.

[6] GReAT, "Wannacry ransomware." [Online]. Available: https://securelist.com/wannacry-ransomware-used-in-widespread-attacks-all-over-the-world/78351/. [Accessed: 20-May-2017].

[7] J. B. Fraley and M. Figueroa, "Polymorphic malware detection using topological feature extraction with data mining," *IEEE SoutheastCon 2016*, pp. 1–7, 2016.

[8] M. Chau, G. Alan Wang, and H. Chen, "A Syntactic Approach for Detecting Viral Polymorphic Malware Variants," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9650, no. April, 2016.

[9] Q. Jiang, "A Feature Selection Method for Malware Detection," *Proceeding of the IEEE International Conference on Information and Automation*, no. June, pp. 890–895, 2011.

[10] C.-T. LIN, "Feature Selection and Extraction for Malware Classification," *Journal of Information Science and Engineering*, vol. 31, pp. 965–992, 2015.

[11] S. Yusirwan, Y. Prayudi, and I. Riadi, "Implementation of Malware Analysis using Static and Dynamic Analysis Method," *International Journal Of Computer Applications*, vol. 117, no. 6, pp. 11–15, 2015.

[12] A. Verma, M. Rao, A. Gupta, W. Jeberson, and V. Singh, "a Literature Review on Malware and Its Analysis," *Int J Cur Res Rev*, vol. 5, no. 16, pp. 71–82, 2013.

[13] J. Landage and M. Wankhade, "Malware and Malware Detection Techniques: A Survey," *International Journal of Engineering Research …*, vol. 2, no. 12, pp. 61–68, 2013.

[14] S. Gadhiya, K. Bhavsar, and P. D. Student, "Techniques for Malware Analysis," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 4, pp. 2277–128, 2013.

[15] R. Kaur and M. Singh, "A survey on zero-day polymorphic worm detection techniques," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 3, pp. 1520–1549, 2014.

[16] Y. Prayudi and S. Yusirwan, "the Recognize of Malware Characteristics Through Static and Dynamic Analysis Approach As an Effort To Prevent Cybercrime Activities," *Journal of Theoretical and Applied Information Technology (JATIT)*, vol. 77, no. xx, pp. 438–445, 2015.

[17] "Malware Classification by Using WEKATOOL," *International Journal of Engineering Science and Computing*, no. July, 2014.

[18] S. K. Pandey and B. M. Mehtre, "A lifecycle based approach for malware analysis," *Proceedings - 2014 4th International Conference on Communication Systems and Network Technologies, CSNT 2014*, pp. 767–771, 2014.

[19] U. Baldangombo, N. Jambaljav, and S.-J. Horng, "A Static Malware Detection System Using Data Mining Methods," *International Journal of Artificial Intelligence & Applications*, vol. 4, no. 4, p. 113, 2013.

[20] R. Islam, R. Tian, L. Batten, and S. Versteeg, "Classification of malware based on string and function feature selection," *Proceedings - 2nd Cybercrime and Trustworthy Computing Workshop, CTC 2010*, pp. 9–17, 2010.

[21] R. Pasha, Y. Prathima, and L. Thirupati, "Malwise System for Packed and Polymorphic Malware," vol. 3, no. 1, pp. 167–172, 2014.

[22] S. Cesare, Y. Xiang, and W. Zhou, "Control flow-based malware variant detection," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 4, pp. 304–317, 2014.

[23] "Malware zoo." [Online]. Available: https://github.com/ytisf/theZoo/tree/master/malwares/Binaries/Potao Express. [Accessed: 02-May-2017].

[24] M. Cantoni, "Malware archives." [Online]. Available: http://www.nothink.org/honeypots/malware-archives. [Accessed: 26-Mar-2016].

[25] M. Bat-Erdene, T. Kim, H. Park, and H. Lee, "Packer Detection for Multi-Layer Executables Using Entropy Analysis," *Entropy*, pp. 1–18, 2017.