

# Malware detection using Gini, Simpson diversity, and Shannon-Wiener indexes

Yeong Tyng Ling<sup>1</sup>, Kang Leng Chiew<sup>1</sup>, Piau Phang<sup>1</sup>, Xiaowei Zhang<sup>2</sup>

<sup>1</sup>Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak, Kota Samarahan, Malaysia

<sup>2</sup>Faculty of Biomedical Engineering, Chengde Medical University, Chengde, China

## Article Info

### Article history:

Received Aug 20, 2024

Revised Nov 19, 2024

Accepted Dec 15, 2024

### Keywords:

Gini coefficient  
Malware detection  
MLP  
Shannon-Wiener  
Simpson diversity  
XGBoost

## ABSTRACT

The increasing number of malware attacks poses a significant challenge to cyber security. This paper proposes a methodology for static malware analysis using biodiversity-inspired metrics that is Gini coefficient, Simpson diversity, and Shannon-Wiener index for malware detection. These metrics are used to build the structural feature representation on the raw binary file as the feature space. The effectiveness of these metrics are evaluated using multilayer perceptron (MLP) neural network and extreme gradient boosting (XGBoost) models. A deterministic algorithm is used to generate these features that represent the feature signature of the executable file. Additionally, we investigated the effectiveness of different byte sizes as the input feature for these two classifiers. According to the results, Gini coefficient with on chunk size of 128 has successfully achieved average F1 score of more than 98.7% by using XGBoost model.

*This is an open access article under the [CC BY-SA](#) license.*



## Corresponding Author:

Yeong Tyng Ling

Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak

Kota Samarahan, Sarawak, Malaysia

Email: ytling@unimas.my

## 1. INTRODUCTION

Malware attack is one of the most significant and prevailing issues in information security. According to [1], there has been an increase of malicious tasks since Q1 2024. Hackers use malicious software to cause harm to a computer or its users in the form of virus, worm, rootkit, key logger, trojan horse, ransomware, and spyware. Traditional commercial anti-malware tools which use signature-based detection method are infamously inefficient when faced with newly launched (a.k.a. “zero-day”) malware. Essentially, this method extracts unique byte sequences which define the malware’s signature in the file contents of previously seen malware. However, this method is time-consuming and costly since it requires newly extracted signatures to be compared against large databases of malicious signatures [2]. It also needs periodic update since malware writers are constantly developing new codes to thwart detection. Hence, advanced protection technology using machine learning (ML) is needed.

There are two methods for malware analysis, dynamic or static analysis. In dynamic analysis [3]–[5], malware features such as runtime API or system call traces are generated by executing a malware file and observing its behavior in a controlled environment, e.g. sandbox, to prevent infection and spreading during analysis. In static analysis [6]–[8], malware features such as n-gram, image representation, opcode are generated without executing the malware file. Table 1 shows the summary of studies by the most current ones related to our work.

There exist several compelling works that used ML models for malware detection and classification. These MLs differ mainly in the algorithms and the types of analysis used. The use of dynamic analysis such as the [3] evaluated eight machines learning algorithms for malware detection through analysis of the frequency of Windows API system function calls. The authors observed the behavior of malware in an isolated environment by using Cuckoo [9]. The behavioral events reported by Cuckoo will be the feature to be fed into the ML models. The authors also applied Gini index in the decision tree model. Similar technique was conducted by Syeda and Asghar [5] where applied both Chi2 and the Gini index to filter and select significant features before being fed into six ML models. For static analysis, studies such as the [6] incorporated word embedding technique on opcode sequence feature with long short-term memory (LSTMs) for malware classification. They used the 30 most frequent opcodes extracted after disassemble the executable files of 20 different malware families. Other studies such as the [7], [8] compared models performance were also been examined for malware classification and had shown promising results by using extreme gradient boosting (XGBoost). Hybrid work in [4] conducted both static and dynamic malware analysis with different ML models. They obtained the best detection accuracy rate of 91.9% on the static analysis dataset and 96.4% on the dynamic analysis dataset by using the XGBoost algorithm. Their study indicates that combining static and dynamic analysis with ML is an effective approach for identifying malware. Their results show that the efficacy of ML model is dependent upon the respective algorithm and the type of data that the model is built upon.

Table 1. Malware detection related studies

Reference	Analysis	Feature	Approach	Dataset (size)	Accuracy
[3]	Dynamic	API	MLs	Malpedia 7400	99.50%
[5]	Dynamic	API	Random forest	MalwareBazaar 582	96.00%
[10]	Static	Aggregation metrics	ELM	APK 600	82.50%
[8]	Static	String	XGBoost	EMBER 5000K	98.50%
[6]	Static	Opcode	LSTM, CNN	Malicia; previous study 25901	81.00%
[7]	Static	Entropy, Gini	MLs, neural network	VirusShare 938	92.17%
[11]	Static	Image	CNNs, ELMs	MalImg 9300	97.70%
[12]	Static	Image	CNN	Malimg 9435	98.82%
[13]	Static	Image	CNN	Malimg 9389	97.32%
[14]	Static	Entropy, image	SNN	Andro-Dumpsys 906	91.20%
[4]	Both	Function, API	XGBoost	VirusShare 2747-2937	96.48%

Lately, deep learning is gaining much popularity due to it's supremacy in terms of accuracy when trained with huge amount of data. Neural networks are a subset of ML, and the heart of deep learning algorithms. There have been many studies that utilized neural networks by adding convolution and pooling layers. The study [12]–[14], used variants of convolutional neural network (CNN) models with image-based and other types of feature representation for malware detection. To overcome the android malware prediction model, [10] studied the patterns of intermediate code and source code of an apk file by extracting 16 types of metrics, such as mean, median, Gini index, and entropy. From their empirical study, extreme learning machine (ELM) with polynomial kernels provides a better performance than other ML classifiers.

Regardless of using ML or deep learning as model classifiers, feature representation plays a crucial role in malware detection. In static analysis, most of the malware come in the form of raw binary executable file. To quantify the raw bytes, [15] introduced the diversity indexes to quantify the qualitative value of malware data. The authors used [16] to compute the different diversity indexes such as Shannon index, Simpson, inverse Simpson, and Fisher's log. Their experimental results show that the ecological metric can be well used in malware context to better understand the pattern in malware. Other studies such as in [17], [18] adopted mathematical models of biodiversity in ecology for detection. Their studies demonstrated that biodiversity-related metrics can improve their understanding of how diversity affects detection.

Inspired by the above related work, in this paper we explore the effectiveness of structural feature of Gini index, Simpson diversity, and Shannon-Wiener index with multilayer perceptron (MLP) neural network and XGBoost models. The rest of this paper is organized as follows. Construction of feature representation of this study is described in section 2. The detailed results and discussion are presented in section 3. Conclusions and suggested future work are discussed in section 4.

## 2. METHOD

In this section, the proposed method is presented. A general flow of the proposed approach is shown in Figure 1.

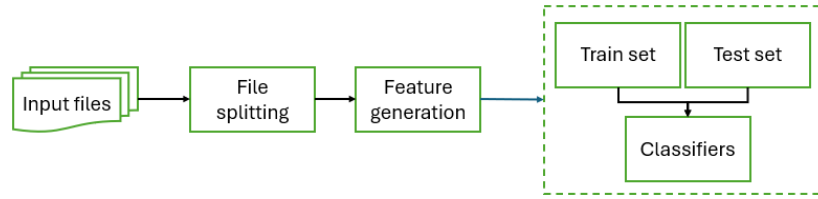


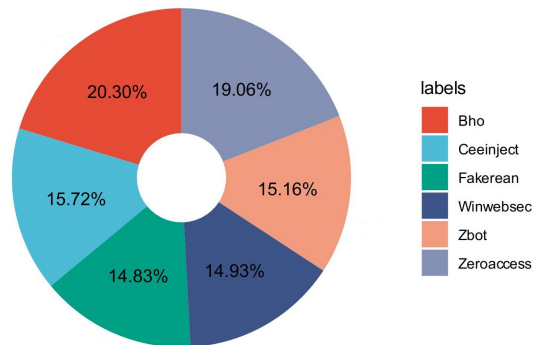
Figure 1. General architecture flow

### 2.1. Input files

The format of the input files used in this study was in Windows executable format. A total of 7,852 binary files were collected as the dataset of the input files. Table 2 lists the information about the malware families used in this study. These malware were collect from [19]. As for the corresponding benign files, we randomly selected 1000 Windows applications with size 0.01~94 MB from <https://download.cnet.com>. Figure 2 shows the distribution of the malware families on the selected dataset.

Table 2. Malware information

Family	Type	Size (MB)	Sample
Bho	Trojan	0.005 - 16.0	1,391
Ceeinject	Virttool	0.004 - 7.67	1,077
Fakerean	Rogue	0.003 - 21.9	1,016
Winwebsec	Rogue	0.325 - 0.60	1,023
Zbot	Trojan	0.031 - 0.37	1,039
Zeroaccess	Trojan	0.048 - 0.28	1,306



Malware Distribution

Figure 2. Malware distribution use in the experiment (for colors)

### 2.2. File splitting

In this step, an input file was split into a series of chunks. To achieve this, technique [20] was adopted for generating the proposed structural feature representation by splitting the entire file into fixed-byte of chunks. A chunk is considered to be a string of non-overlapping consecutive bytes, where each chunk contains the same number of bytes. To do this, a unanimous file length,  $F$ , and, therefore, the number of chunks,  $N$ , have to be determined. We fixed the file length to be a power of 2, i.e.  $N = 2^\alpha$  for some  $\alpha \in N$ :

$$\alpha = \left\lceil \log \frac{\min\{\text{median}(M), \text{median}(B)\}}{c} \right\rceil$$

For convenience purposes, steps to determine  $\alpha$  are restated here as follows:

- Step 1: compute the median size of a group of malware files and benign files,  $M$  and  $B$ , respectively. Here, differ from [20], median score is considered as it usually provides a better measure of center tendency of sample size.
- Step 2: determine the minimum median size from these two groups.
- Step 3: divide the minimum median size by chunk size, says  $c = 256$  bytes, this gives the  $D$ .
- Step 4: find the base-2 logarithm of value from previous step and take the largest whole integer.
- Step 5: if the whole integer in previous step is not a power of 2, reduce the  $D$  in step 3 by 1 and repeat step 4 until the condition met.

Different chunk sizes, that is, 128, 256, 512, 1,024, and 2,048 bytes were examined in this study. The sliding window for each file splitting is the same length as the chunk sizes for convenient purpose. These chunks provide granular variations and represent the structure of a file.

### 2.3. Feature generation

Based on [20], once the number of chunks,  $N$ , has been determined, a deterministic algorithm using Procrustean notion is adopted to choose evenly spread chunks from each file to produce a vector of  $N$  chunks in order. In other words, the number of chunks for a file is either reduced to or increased to  $N$ . An example is provided here for illustration purpose. Given two files,  $P$  and  $Q$ , and a chunk size of  $c$ , with  $length(P) = 10c$  and  $length(Q) = 7c$  which means there are 20 chunks for  $P$  and 6 chunks for  $Q$ . Suppose that  $\alpha = 3$  is chosen, then  $N = 2^\alpha = 8$  chunks. Since  $P$  has number of chunks larger than  $N$ , it needed to be reduced from 10 to 8 chunks and for  $Q$  which is smaller than  $N$ , it needed to be increased from 7 to 8 chunks. In order to choose these chunks, a subset of the current chunks using a jump factor is generated for each file. The chunk index is initially set to 0, and it is incremented in every step by  $inc_1 = 9/7 = 1.28$  for  $P$  and  $inc_2 = 6/7 = 0.85$  for  $Q$ . The indices are selected using the floor of the accumulated jump value, so the chosen indices will be:

$$I_P = (0, 1, 2, 3, 5, 6, 7, 9) \quad I_Q = (0, 0, 1, 2, 3, 4, 5, 6)$$

These indices give the location of chunk that needed to build the structural feature representation of a file. Three biodiversity-inspired metrics were adopted in this study to build the structural feature representation on these chunks, namely, Gini coefficient, Simpson diversity, and Shannon-Wiener.

#### 2.3.1. Gini coefficient

Gini coefficient also known as Gini index [21], named after Italian statistician Corrado Gini, is a way to measure statistical dispersion inequality especially in economics and ecology [22]. The Gini coefficient is defined as:

$$gini = t / (b^2 * a) \quad (1)$$

where  $t$  is a list of difference among the elements of a list,  $b$  is the length of a list and  $a$  is mean value of that list.

#### 2.3.2. Simpson diversity

The Simpson diversity index [23] was introduced by Edward H. Simpson to measure the probability of two samples will belong to the same group. The value of Simpson diversity ranges from 0 to 1, with 0 representing large diversity and 1 representing no diversity. The formula is given as:

$$D = \sum_{i=1}^R \left( \frac{n_i(n-1)}{N(N-1)} \right) \quad (2)$$

where  $n_i$  is the number of individuals in a group  $i$ , and  $N$  is the total number of groups in a sample.

### 2.3.3. Shannon-Wiener

The Shannon-Weiner index [24] was developed from information theory and is based on measuring uncertainty. The computational formula is:

$$H' = \frac{N \ln N - \sum (n_i \ln n_i)}{N} \quad (3)$$

where  $N$  is the total number of groups and  $n_i$  is the number of individuals in group  $i$ . Each metric, which is used to quantize the byte randomness in a chunk, will define the  $N$ -vector structural feature of a file. Thus, for a given executable file, three different types of structural features were generated.

## 2.4. Classifiers

Next, each type of the structural feature generated from the previous step will be fed into two selected classifiers, namely MLP neural network and XGBoost, respectively. All the experiments were conducted under a Windows 10 64-bit operating system using the Python programming language (Scikit-learn library). A 10-fold cross validation was performed to estimate the generalization performance of the proposed approach. The prediction efficiency was measured in terms of accuracy rate, area under curve (AUC), and F1 score.

### 2.4.1. Multilayer perceptron

Based on the layer construction of MLP by [25], a 3-layer MLP model is constructed as having one input layer, two hidden layers, followed by a output layer. After several experiments using 3 and 4 layers, we decided to use the 3-layer MLP model as it gave better performance than the 4-layer model. In this study, the first layer (input layer with linear layer), with  $N$  nodes, where  $N$  is the number of chunks generated during the file splitting step. Activation function was set to the rectified linear unit (ReLU) to suppresses negative weights. This process is repeated for another hidden layer by reducing the half of the previous nodes. The last layer is a Sigmoid curve which allows binary prediction. Additionally, dropout layers with 0.2 is added between hidden layers to prevent overfitting. The model was fed through and backpropagate of errors with 50 epochs. The learning rate is 0.01 with binary cross entropy (BCE) as the loss function and stochastic gradient descent as the optimizer.

### 2.4.2. XGBoost

XGBoost [26], is an implementation of gradient boosted decision trees (GBDTs). The XGBoost provides a wrapper class to allow models to be treated like classifiers in the scikit-learn framework in Python. There are many parameters for the XGBoost Classifier package. We kept them as in the default for simplicity reasons and only set the *objective*='binary:logistic'. A 10-fold cross validation was performed on different chunk sizes (i.e.: 128) of the proposed structural feature representation as mentioned in section 2.3.

## 3. RESULTS AND DISCUSSION

We want to study how effective are the proposed structural features in discriminating malware from benign files in terms of accuracy, AUC, and F1 score. We consider a validation result of at least 90% as high detection rate. Based on the findings, a discussion section is followed.

### 3.1. Results

Figures 3 shows the average time (in seconds) taken to generate chunk size of 128 and 2,048 bytes of the proposed structural feature representations for the malware family. It is obvious that to extract smaller chunk size, says 128 bytes, will take longer time compare to chunk size of 2,048 bytes. Based on the figure, it can be observed that using the Simpson index to generate the structural feature is the fastest, followed by Gini coefficient and Shannon-Weiner. It is surprising to notice that the time taken by Winwebsec family is longer than the other families, such as the Bho which contains more samples than Winwebsec when using Gini coefficient and Shannon-Weiner. One possible conjecture is that the Winwebsec family has file sizes that is much larger than the other families.

#### 3.1.1. MLP model

Table 3 shows the comparison of the best F1 score performance out of the 10-fold cross validation. The highlighted bold indicates the highest score achieved among the six malware families on respective chunk size. It is observed that Gini coefficient produced the highest F1 scores on Winwebsec family except with

chunk size 256. As with chunk size 128, 512, 1,024, and 2,048 the MLP model can achieve 99.84%, 99.32%, 99.17%, and 98.75% F1 scores on Winwebsec, respectively.

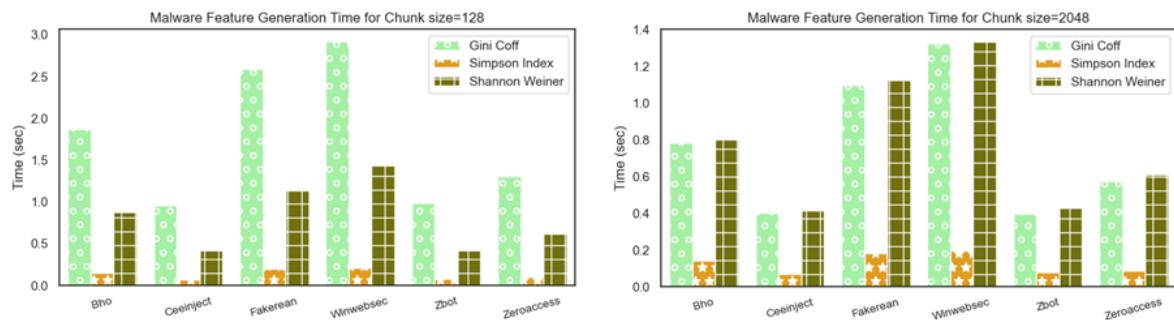


Figure 3. Average time of feature generation of the malware family

Table 3. The best F1 score using the MLP model

Chunk size	Family	Feature type		
		Gini coefficient	Simpson diversity	Shannon Wiener
128	Bho	87.51	89.19	88.73
	Ceeinject	89.37	90.27	92.23
	Fakerean	87.86	90.44	89.62
	Winwebsec	<b>99.84</b>	99.00	98.89
	Zbot	96.93	98.07	97.72
	Zeroaccess	99.22	99.49	97.71
256	Bho	87.32	89.35	86.10
	Ceeinject	89.15	90.41	90.60
	Fakerean	86.07	91.53	86.34
	Winwebsec	99.36	<b>99.52</b>	98.73
	Zbot	98.11	98.49	94.90
	Zeroaccess	98.73	99.25	96.83
512	Bho	89.24	89.26	83.44
	Ceeinject	88.52	87.77	87.36
	Fakerean	90.85	91.09	83.24
	Winwebsec	<b>99.32</b>	99.21	98.05
	Zbot	96.84	98.13	92.04
	Zeroaccess	98.57	98.85	91.55
1024	Bho	88.30	87.66	81.50
	Ceeinject	84.53	85.46	80.26
	Fakerean	90.06	91.00	81.69
	Winwebsec	<b>99.17</b>	98.85	95.45
	Zbot	96.78	97.52	90.55
	Zeroaccess	97.81	98.08	85.39
2048	Bho	87.21	89.35	78.73
	Ceeinject	86.07	84.98	64.07
	Fakerean	90.63	92.15	83.75
	Winwebsec	<b>98.75</b>	97.99	90.59
	Zbot	97.11	97.00	85.40
	Zeroaccess	96.87	97.86	79.95

Figure 4 depicts a closer look at the performance between two selected malware families, i.e.: Winwebsec and Bho. Based on the figure, the Winwebsec family in Figure 4(a) can easily be detected compared with the Bho family in Figure 4(b) using Gini coefficient as feature representation. The Simpson diversity index achieved stable performance across all chunk sizes. Table 4 shows the accuracy rate based on the best F1 score achieved across the malware families. It measures how often the MLP model correctly predicts the outcome. As the chunk size grows larger, the F1 score decreases by using Shannon-Wiener as the structural feature representation.

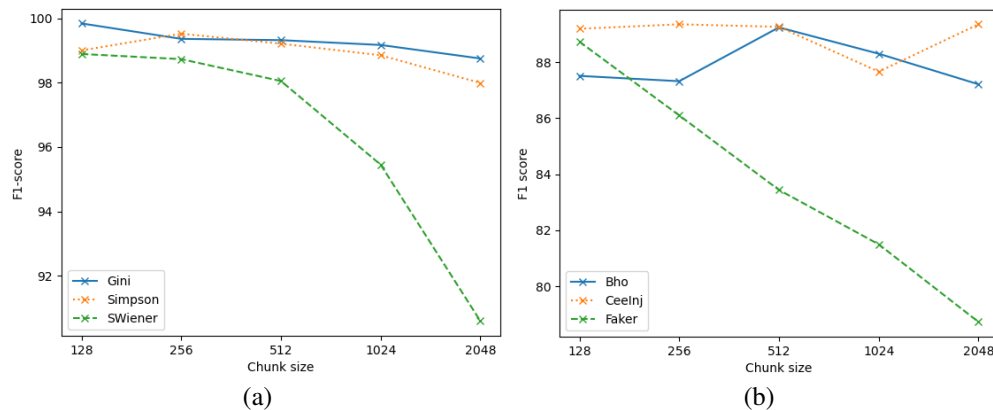


Figure 4. The F1 score for (a) Winwebsec and (b) Bho families (see online version for colors)

Table 4. The accuracy rate based on the best F1 score

Chunk size	Family	Feature type		
		Gini coefficient	Simpson diversity	Shannon Wiener
128	Bho	84.54	87.04	86.76
	Ceeinject	89.08	89.24	91.81
	Fakerean	87.76	90.08	89.09
	Winwebsec	<b>99.83</b>	99.01	98.84
	Zbot	97.05	98.03	97.71
	Zeroaccess	99.13	99.42	97.25
256	Bho	87.32	86.62	82.86
	Ceeinject	88.44	89.88	90.20
	Fakerean	85.45	91.40	86.66
	Winwebsec	99.34	<b>99.50</b>	98.68
	Zbot	98.03	98.36	94.93
	Zeroaccess	98.55	99.13	96.53
512	Bho	86.90	87.46	82.31
	Ceeinject	87.64	87.47	86.99
	Fakerean	96.41	91.23	84.29
	Winwebsec	<b>99.34</b>	99.17	98.02
	Zbot	96.73	98.03	91.50
	Zeroaccess	98.41	98.29	89.73
1024	Bho	86.35	85.65	78.83
	Ceeinject	83.78	84.10	80.73
	Fakerean	89.42	91.23	84.95
	Winwebsec	<b>99.17</b>	98.84	95.38
	Zbot	96.73	97.38	90.52
	Zeroaccess	97.39	97.83	80.92
2048	Bho	84.81	88.02	71.86
	Ceeinject	85.87	84.91	70.30
	Fakerean	90.74	92.06	84.01
	Winwebsec	<b>98.68</b>	98.02	90.28
	Zbot	96.89	96.73	84.64
	Zeroaccess	96.38	97.68	72.39

It is observed that the accuracy rates are consistent with the F1 score, where using Gini coefficient can achieve the highest accuracy rate for the Winwebsec family except with chunk size 256. Among the six families, Gini coefficient can effectively detect the malware from the benign file for the Winwebsec, Zbot, and Zeroaccess. It implies that these three malware families can easily be detected using this Gini coefficient, but not for the Bho, Ceeinject, and Fakerean when compared with the other two structural feature representations.

On average, feature representation using Simpson diversity shown relatively higher accuracy rate than the Gini coefficient across all the malware families. A closer observation shows that this structural feature representation can achieve more than 90% for four of the families except for the Bho and Ceeinject families.

This shows that Simpson diversity demonstrated stronger discrimination for quantifying byte information.

Shannon-Wiener shown as the least significant structural feature representation in this study. The lowest accuracy rate it can yield is 71.86% for the Bho family with chunk size 2,048 and the highest accuracy rate it can yield is 98.68% for the Winwebsec family with chunk size 256. One can observe that as the chunk size increases, the discrimination power for this feature representation becomes worse.

Figure 5 depicts the AUC performance based on the three proposed structural feature representations in Figures 5(a) to 5(c). AUC represents the degree or measure of separability. It can be observed that there is a clear distinction between AUC for certain types of malware families. For example, either both Gini coefficient and Simpson index yielded higher AUC for the Winwebsec, Zbot, and Zeroaccess, but not for the Bho, Ceeinject, and Fakerean families. The performance of all three structural feature representations declines as the number of chunk sizes increases.

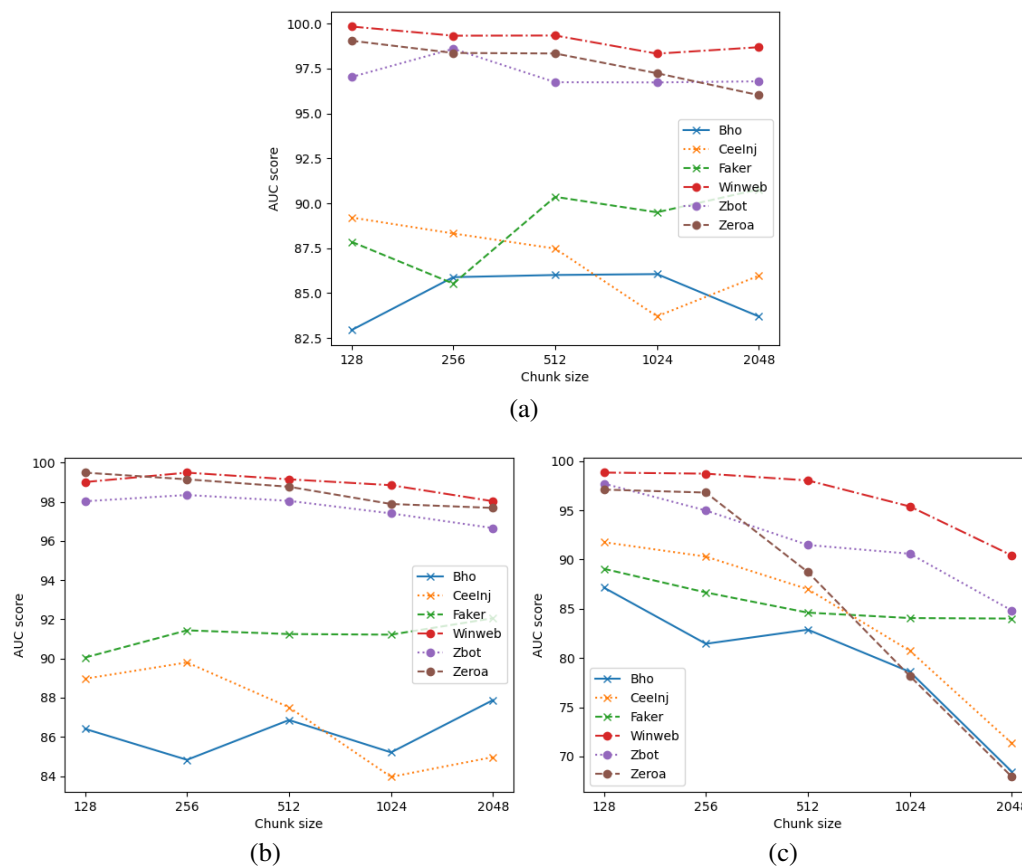


Figure 5. AUC performance: (a) Gini coefficient, (b) Simpson index, and (c) Shannon Wiener

### 3.1.2. XGBoost model

Table 5 shows the comparison of the best F1 score performance out of the 10-fold cross validation. The highlighted bold indicates the highest score achieved on respective chunk size. It is observed that the average precision and recall varies for each of the feature representation. By using the XGBoost model, Winwebsec family achieved 100% F1 score with all the three proposed structural feature representations across all different chunk sizes. Zeroaccess family, similar to the Winwebsec family, also reached F1 score of 100% for all the chunk sizes except with chunk size 1,024.

In terms of feature performance, on average, Shannon-Wiener achieved more number of highest F1 score followed by Gini coefficient and Simpson diversity. However, the average F1 score is 98.26%. It is followed by Gini coefficient and Simpson diversity, which yielded an average F1 score of 98.63%, 98.27% respectively. In terms of chunk size, Shannon-Wiener and Simpson diversity demonstrated their discriminate



power mostly with chunk size 256 and 512, respectively. As for Gini coefficient, it demonstrated its discriminate power mostly with chunk size 128 and 1,024.

Figure 6 depicts a closer look at the performance between two malware families, i.e.: Winwebsec and Bho. Based on the figure, the Winwebsec family in Figure 6(a) can also be easily detected compared with Bho family in Figure 6(b) using all the three proposed structural feature representations. However, for the Bho family, Shannon-Wiener index performed best with chunk size 256 and 1,024 only. Both the Gini coefficient and Simpson index produced best result with chunk size 512 and 2,048, respectively.

Table 5. The F1 score of the best model performance using the XGBoost model

Chunk size	Family	Feature type		
		Gini coefficient	Simpson diversity	Shannon Wiener
128	Bho	95.53	94.94	95.20
	Ceeinject	98.24	97.52	97.41
	Fakerean	99.09	98.13	97.69
	Winwebsec	100.0	100.0	100.0
	Zbot	99.54	99.53	<b>100.0</b>
	Zeroaccess	100.0	100.0	100.0
256	Bho	96.24	96.84	97.57
	Ceeinject	96.96	97.41	96.96
	Fakerean	98.21	97.32	95.32
	Winwebsec	100.0	100.0	100.0
	Zbot	99.09	99.50	<b>100.0</b>
	Zeroaccess	100.0	100.0	100.0
512	Bho	97.27	97.14	96.86
	Ceeinject	95.37	95.85	95.85
	Fakerean	97.32	97.67	97.75
	Winwebsec	100.0	100.0	100.0
	Zbot	99.54	<b>100.0</b>	99.50
	Zeroaccess	100.0	100.0	100.0
1024	Bho	96.86	96.50	96.88
	Ceeinject	95.39	96.10	94.68
	Fakerean	98.65	97.65	98.21
	Winwebsec	100.0	100.0	100.0
	Zbot	99.06	99.38	99.50
	Zeroaccess	<b>100.0</b>	99.28	99.63
2048	Bho	96.00	97.16	96.52
	Ceeinject	94.49	93.10	94.82
	Fakerean	98.64	97.65	98.18
	Winwebsec	100.0	100.0	100.0
	Zbot	99.09	99.49	99.50
	Zeroaccess	99.59	<b>100.0</b>	<b>100.0</b>

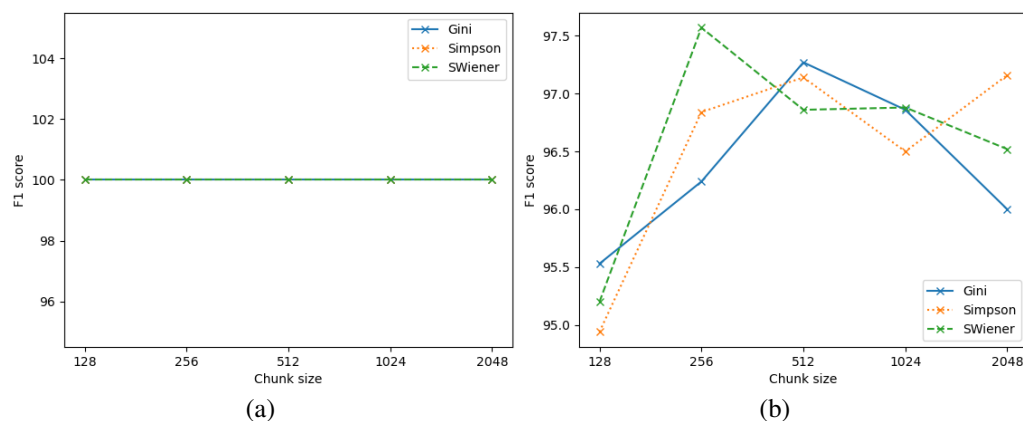


Figure 6. F1 score of comparison between (a) Winwebsec and (b) Bho families (see online version for colors)

Table 6 shows the accuracy rate based on the best F1 score across the malware families. The highlighted bold indicates the significant rate achieved among the malware families on respective chunk size. It is observed that both Shannon-Wiener and Gini coefficient have the most number of times to yield higher accuracy rates across the malware families.

Figure 7 depicts the AUC performance based on the three structural feature representations as shown in Figures 7(a) to 7(c). Here, it is clear that the Gini coefficient produced higher AUC for most of the chunk sizes except with chunk size 512. On the other hand, Simpson diversity and Shannon-Wiener features performed better with chunk size 512. Shannon-Wiener can yield high detection the Winwebsec, Zbot, and Zeroaccess families with chunk size 2,048, that is, 100%, 99.01%, and 100%, respectively.

Table 6. The accuracy rate of the best model based on the F1 score

Chunk size	Family	Feature type		
		Gini coefficient	Simpson diversity	Shannon Wiener
128	Bho	95.53	94.94	95.20
	Ceeinject	98.24	97.52	97.41
	Fakerean	99.09	98.13	97.69
	Winwebsec	100.0	100.0	100.0
	Zbot	99.54	99.53	<b>100.0</b>
	Zeroaccess	100.0	100.0	100.0
256	Bho	96.24	96.84	97.57
	Ceeinject	96.96	97.41	96.96
	Fakerean	98.21	97.32	95.32
	Winwebsec	98.21	97.32	95.32
	Zbot	100.0	100.0	100.0
	Zeroaccess	99.09	99.50	<b>100.0</b>
512	Bho	97.27	97.14	96.86
	Ceeinject	95.37	95.85	95.85
	Fakerean	97.32	97.67	97.75
	Winwebsec	100.0	100.0	100.0
	Zbot	99.54	<b>100.0</b>	99.50
	Zeroaccess	100.0	100.0	100.0
1024	Bho	96.86	96.50	96.88
	Ceeinject	95.39	96.10	94.68
	Fakerean	98.65	97.65	98.21
	Winwebsec	100.0	100.0	100.0
	Zbot	96.06	99.38	99.50
	Zeroaccess	<b>100.0</b>	99.28	99.63
2048	Bho	96.00	97.16	96.52
	Ceeinject	94.49	93.10	94.82
	Fakerean	98.64	97.65	98.18
	Winwebsec	100.0	100.0	100.0
	Zbot	99.09	99.49	99.50
	Zeroaccess	99.59	<b>100.0</b>	<b>100.0</b>

### 3.2. Discussion

While earlier studies have explored the impact of biodiversity-related metrics, they have not explicitly studied their effectiveness for quantifying on the binary file. This study investigated the effectiveness of three biodiversity-related metrics, namely Gini coefficient, Simpson diversity, and Shannon-Wiener, on binary file for malware detection. The validation results from our study suggests that the computation steps to extract and generate structural feature representation shall be considered if the performance speed is a concern. In this study, the number of large files in the Winwebsec family may be contributing to the fact that it required more feature generation time than the Bho family. Due to the computational steps in the Gini coefficient, it takes longer time to generate the structural feature of a file.

Comparing the three structural feature representations based on the F1 score, our study suggests that Gini coefficient with XGBoost can be an effective metric to quantify binary file for detection on average. This may be due to the fact that this metric measures the probability for a value within a chunk bytes and the type of malware family can also affect the performance. It is unclear the reason why Shannon-Wiener index produces low performance in the experiment here. It is suspected that the computation of  $\ln$  causes the diversity value

too low that can not be distinguished with the benign file. However, further and in-depth studies may be needed to confirm this.

There are some strengths of the approach proposed in this study. In general, binary static analysis has the advantage over the disassembling process in terms of processing cost. By capturing the fine-grain structure of a file, this approach is able to inspect the entire file without actually executing it. Furthermore, by adopting the advancements of ML algorithms, the proposed approach has the advantages in yielding effective results with computing efficiency. By exploring different ways to quantize byte data, this approach is able to transform randomness information in a byte chunk into feature representation for distinguish malware from benign files. There exist some limitations of the present study. First, the need to extract and generate feature representation may incur issue of effectiveness in zero-day attack. Second, MLP can require significant computational resources such as memory and processing power. Third, there is no feature space reduction was conducted in this study. By imposing a feature reduction step, the overall performance of both models can be studied, especially the effect of the reduced feature space to double confirm the findings.

This study also explored a comprehensive study on the performance of two classifiers, namely MLP and XGBoost. Our findings provide conclusive evidence that biodiversity-inspired metrics with ML models is effective for malware detection. Based on the results, XGBoost has higher detection rate compare to MLP. These results further support the common understanding that XGBoost, a gradient boosting framework, excel in tabular dataset. For both MLP and XGBoost models, Bho and Ceeinject are the malware families that show difficult to be detected.

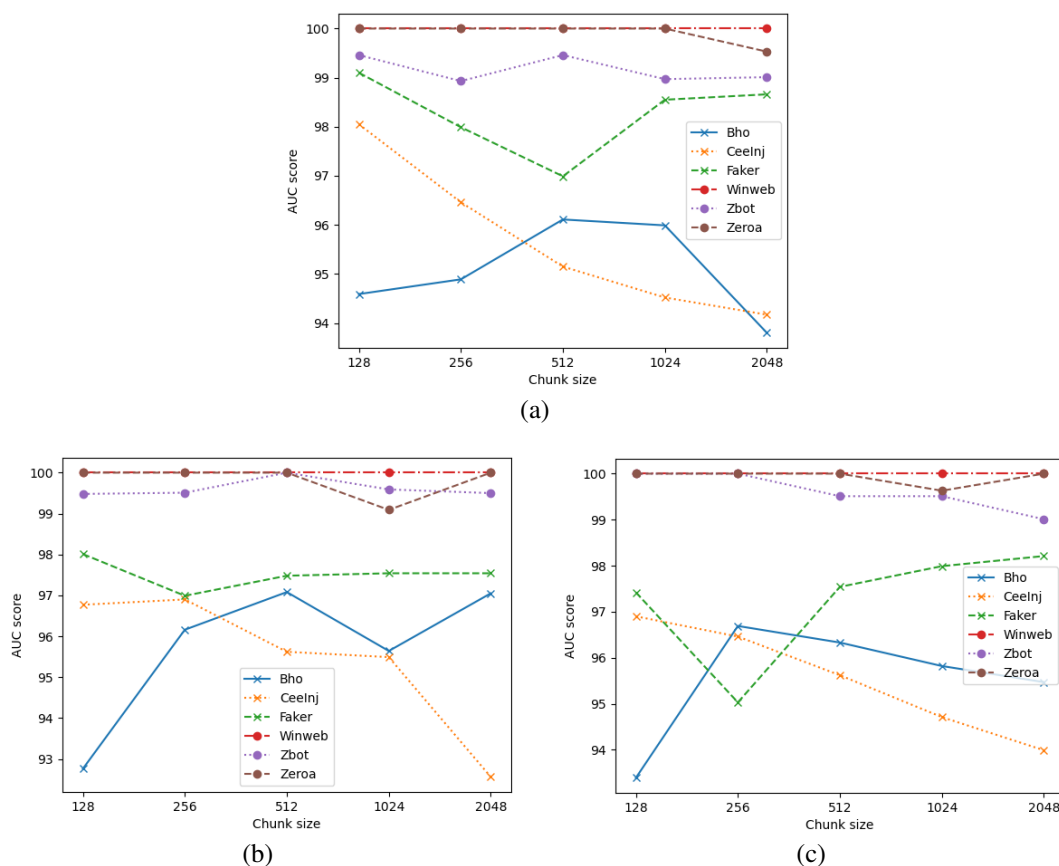


Figure 7. AUC performance: (a) Gini coefficient, (b) Simpson index, and (c) Shannon Wiener

#### 4. CONCLUSION

This paper presented an approach of malware detection using metrics from ecology domain to construct structural feature representation. Namely, Gini coefficient, Simpson diversity, and Shannon-Wiener

indexes were used to quantify and enfold the raw bytes as feature for binary executable file. The proposed structural features were successfully evaluated to achieve average F1 score more than 98.7% by using Gini coefficient with XGBoost on chunk size of 128, which was better than using MLP model. Gini coefficient and Shannon-Wiener both achieved accuracy rate of 95-100%, AUC of 93-100%, and F1 score of 95-100% with chunk size of 128. By exploring the effectiveness of biodiversity-inspired metrics in quantifying byte information, it builds an alternative approach for better malware detection. This research work can be followed up in several different directions. First, it would be significant to study the effectiveness of structural features on different types of ensemble learning models. Second, exploring cryptography technique in quantifying the raw bytes as feature representation. Last, dimension reduction techniques can be considered to reduce the structural feature space and reduce complexity during the pre-processing stage.

## ACKNOWLEDGEMENTS

This research is supported by Universiti Malaysia Sarawak under grant F08/PARTNERS/2108/2021.




## REFERENCES

- [1] "Malware trends report: Q2, 2024," *Any Run*, 2024. <https://any.run/cybersecurity-blog/malware-trends-q2-2024/> (accessed Jul. 11, 2024).
- [2] A. Lakhotia, A. Kapoor, and E. Kumar, "Are metamorphic viruses really invincible," *Virus Bulletin*, no. December 2004, pp. 5–7, 2004.
- [3] A. Walker and S. Sengupta, "Insights into malware detection via behavioral frequency analysis using machine learning," in *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, Nov. 2019, vol. 2019-Novem, pp. 1–6, doi: 10.1109/MILCOM47813.2019.9021034.
- [4] J. Palša *et al.*, "MLMD—a malware-detecting antivirus tool based on the XGBoost machine learning algorithm," *Applied Sciences (Switzerland)*, vol. 12, no. 13, p. 6672, 2022, doi: 10.3390/app12136672.
- [5] D. Z. Syeda and M. N. Asghar, "Dynamic malware classification and API categorisation of windows portable executable files using machine learning," *Applied Sciences (Switzerland)*, vol. 14, no. 3, p. 1015, 2024, doi: 10.3390/app14031015.
- [6] D. Dang, F. Di Troia, and M. Stamp, "Malware classification using long short-term memory models," in *Proceedings of the 7th International Conference on Information Systems Security and Privacy*, Mar. 2021, pp. 743–752, doi: 10.5220/0010378007430752.
- [7] D. Baig, M. U. S. Khan, D. Dancey, A. Abbas, M. Ali, and R. Nawaz, "Malware detection and classification along with trade-off analysis for number of features, feature types, and speed," in *2021 International Conference on Frontiers of Information Technology (FIT)*, Dec. 2021, pp. 246–251, doi: 10.1109/FIT53504.2021.00053.
- [8] R. Kumar and S. Geetha, "Malware classification using XGboost-gradient boosted decision tree," *Advances in Science, Technology and Engineering Systems*, vol. 5, no. 5, pp. 536–549, 2020, doi: 10.25046/AJ050566.
- [9] "Cuckoo Sandbox Book," *Cuckoo Foundation*. <https://cuckoo.sh/docs/> (accessed Aug. 15, 2024).
- [10] L. Kumar, C. Hota, A. Mahindru, and L. B. M. Neti, "Android malware prediction using extreme learning machine with different Kernel functions," in *Proceedings of the Asian Internet Engineering Conference on - AINTEC '19*, 2019, pp. 33–40, doi: 10.1145/3340422.3343639.
- [11] M. Jain, W. Andreopoulos, and M. Stamp, "Convolutional neural networks and extreme learning machines for malware classification," *Journal of Computer Virology and Hacking Techniques*, vol. 16, no. 3, pp. 229–244, 2020, doi: 10.1007/s11416-020-00354-y.
- [12] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, "IMCFN: image-based malware classification using fine-tuned convolutional neural network architecture," *Computer Networks*, vol. 171, p. 107138, 2020, doi: 10.1016/j.comnet.2020.107138.
- [13] M. J. Awan *et al.*, "Image-based malware classification using VGG19 network and spatial convolutional attention," *Electronics (Switzerland)*, vol. 10, no. 19, p. 2444, 2021, doi: 10.3390/electronics10192444.
- [14] J. Zhu, J. Jang-Jaccard, A. Singh, P. A. Watters, and S. Camtepe, "Task-aware meta learning-based siamese neural network for classifying control flow obfuscated malware," *Future Internet*, vol. 15, no. 6, p. 214, 2023, doi: 10.3390/fi15060214.
- [15] S. Chamotra, R. K. Sehgal, R. Kamal, and J. S. Bhatia, "Data diversity of a distributed honey net based malware collection system," in *2011 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*, Apr. 2011, pp. 125–129, doi: 10.1109/ETNCC.2011.5958500.
- [16] "OptimaAI suite from R System," *R System*. <https://www.rsystems.com/> (accessed Jul. 11, 2024).
- [17] M. Zhang, L. Wang, S. Jajodia, A. Singhal, and M. Albanese, "Network diversity: a security metric for evaluating the resilience of networks against zero-day attacks," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 1071–1086, 2016, doi: 10.1109/TIFS.2016.2516916.
- [18] M. A. Bouke, A. Abdullah, J. Frnda, K. Cengiz, and B. Salah, "BukaGini: a stability-aware gini index feature selection algorithm for robust model performance," *IEEE Access*, vol. 11, pp. 59386–59396, 2023, doi: 10.1109/ACCESS.2023.3284975.
- [19] S. Basole, F. Di Troia, and M. Stamp, "Multifamily malware models," *Journal of Computer Virology and Hacking Techniques*, vol. 16, no. 1, pp. 79–92, 2020, doi: 10.1007/s11416-019-00345-8.
- [20] H. D. Menéndez, S. Bhattacharya, D. Clark, and E. T. Barr, "The arms race: adversarial search defeats entropy used to detect malware," *Expert Systems with Applications*, vol. 118, pp. 246–260, 2019, doi: 10.1016/j.eswa.2018.10.011.
- [21] C. Gini, "On the measure of concentration with special reference to income and statistics," *Colorado College Publication, Colorado Springs, General Series*, vol. 208, no. 1, pp. 73–79, 1936.
- [22] M. Catalano, T. Leise, and T. Pfaff, "Measuring resource inequality: the gini coefficient," *Numeracy*, vol. 2, no. 2, p. 4, 2009, doi: 10.5038/1936-4660.2.2.4.
- [23] E. H. Simpson, "Measurement of diversity," *Nature*, vol. 163, no. 4148, p. 688, 1949, doi: 10.1038/163688a0.




- [24] B. V. Barnes, D. R. Zak, S. R. Denton, and S. H. Spurr, *Forest ecology*, 4th ed. New York: John Wiley & Sons, Inc, 1998.
- [25] H. T. Ziboon and A. A. Thabit, "A new proposed adaptive cognitive radio detection system based on MLP neural network for different modulation schemes," *ARPJ Journal of Engineering and Applied Sciences*, vol. 12, no. 2, pp. 521–527, 2017.
- [26] T. Chen and C. Guestrin, "XGBoost: a scalable tree boosting system," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, vol. 13-17-August-2016, pp. 785–794, doi: 10.1145/2939672.2939785.

## BIOGRAPHIES OF AUTHORS






**Yeong Tyng Ling**    is a senior lecturer in the Faculty of Computer Science and Information Technology at Universiti Malaysia Sarawak. received her Doctor of Philosophy (Ph.D.) from Universiti Putra Malaysia (UPM) in 2022. Her research interests lie in the area of deep learning, malware detection, parallel computing, and the internet of things (IoT). She can be contacted at email: ytling@unimas.my.






**Kang Leng Chiew**    is currently an associate professor at the Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak (UNIMAS). He received his Ph.D. in computer science specialised in Information Hiding from Macquarie University, Sydney, Australia. His research interest is in information security. He is currently working in anti-phishing research. Past researches include steganalysis on digital images. Previously, he also worked in the area of image processing. He can be contacted at email: klchiew@unimas.my.



**Piau Phang**    is a senior lecturer in the Faculty of Computer Science and Information Technology in Universiti Malaysia Sarawak (UNIMAS). He received his Doctor of Philosophy (Ph.D.) in mathematics in Curtin University of Technology, Perth, Australia at year 2017. He had more than 14 years' teaching experience in undergraduate level. His current research interests include mathematical and statistical epidemiology, ordinary differential equations and game theory. He can be contacted at email: pphang@unimas.my.



**Xiaowei Zhang**    received his Doctor of Philosophy (Ph.D.) degree from Universiti Putra Malaysia in 2024. His research interest is network computing. He is currently a lecturer at the Computer Teaching and Research Section of Biomedical Engineering Faculty, Chengde Medical College (CDMC), China. He can be contacted at email: zxw80000@cdmc.edu.cn.