LETTER *Special Section on Cryptography and Information Security*

# Packer Identification Method for Multi-Layer Executables Using Entropy Analysis with $k$-Nearest Neighbor Algorithm

**Ryoto OMACHI**[†], ***Nonmember and*** **Yasuyuki MURAKAMI**[†a)], ***Member***

**SUMMARY** The damage cost caused by malware has been increasing in the world. Usually, malwares are packed so that it is not detected. It is a hard task even for professional malware analysts to identify the packers especially when the malwares are multi-layer packed. In this letter, we propose a method to identify the packers for multi-layer packed malwares by using $k$-nearest neighbor algorithm with entropy-analysis for the malwares.

***key words:*** *malware, packer identification, multi-layer packing, k-nearest neighbor algorithm, entropy analysis*

## 1. Introduction

The malware attack is known to be one of the most popular cyberattacks. The damage cost caused by malware has been increasing in the world. Thus, reducing the damage cost is an urgent issue.

Malware analysts must analyze the original code and develop anti-malware softwares in order to protect personal data from malwares. If the original code cannot be analyzed, it is difficult to reduce the incidents. Malwares are made by various anti-analysis techniques so as not to be detected by anti-malware softwares.

Packing is an obfuscation method by compressing or encrypting codes in order to hide the original code. A software for packing malwares is called a 'packer'. Recovering the original code from the packed code is called 'unpacking.' In order to analyze the malware, it is necessary to identify the packer and unpack into the original code. It is hard task even for professional malware analysts to identify the packers from multi-packed malwares.

The methods for identifying packers are roughly divided into the following two methods. One is a method of detecting OEP (Original Entry Point) before packing by comparing the code with the run-time library [1]. This method is mainstream, however, there exist following problems. Malware writers usually change the entry point for jamming. It assumes that malware writers used a well-known compiler. It apply only to single-packed malwares. The other is a method of analyzing entropy without running malwares. The former is highly versatile and automatically unpack the malwares for single-packing The latter is relatively safer, but inferior in accuracy because the entropy is made higher by pack.

Manuscript received March 13, 2022.
Manuscript revised June 29, 2022.
Manuscript publicized August 16, 2022.

[†]The authors are with the Osaka Electro-Communication University, Neyagawa-shi, 572-8530 Japan.
a) E-mail: yasuyuki@oecu.jp

DOI: 10.1587/transfun.2022CIL0002

There are three related researches on this letter.

Lyda et al. proposed a method of determining whether the sample is a malware or not by using entropy analysis [2]. Since it can be considered that the characteristics of the executable file appear in the entropy, malwares might be identified by analyzing the entropy. The advantage of entropy analysis is that it is safe because users don't have to run the executables.

Sun et al. proposed a method to detect the single-packer by pattern recognition techniques with $k$-nearest neighbor, etc. [3]. From this result, we can consider that the $k$-nearest neighbor algorithm might be also effective to detect packer of multi-layer packing. However, there are no studies that have performed entropy analysis and $k$-nearest neighbor algorithms in the packer identification from multi-packed malware.

Bat-Erdene et al. proposed a packer detection method for multi-layer packed codes by using symbolic aggregate approximation (SAX), which is a method of converting time-series data into string data [4]. From this result, we consider that SAX method is very useful and good accuracy in mupti-packed malware analysis.

In this letter, we shall propose the method of identifying the packers from multi-layer packed malwares by using SAX method for the entropy vectors with $k$-nearest neighbor method of pattern recognition. We downloaded the executable pseudo malware files provided by eagle0wl [5] and create double-packed malwares from them. Moreover, we shall identify the packers from the double-packed malwares by computer experiment so that we confirm the effectiveness of the proposed method.

## 2. Preliminary

### 2.1 Packing

Packing is a file-conversion method to compress, encode, and obfuscate a program while being kept executable.

There are three types of Packing: single-packing, re-packing and multi-layer packing. Let $m$ be an executable file, $c$ be a packed file and $p_1, p_2, \ldots, p_n$ be packing algorithms.

**single-packing**
$c = p_1(m)$.
**re-packing**
$c = p_1(\ldots(p_1(m))\ldots)$.
**multi-layer packing**

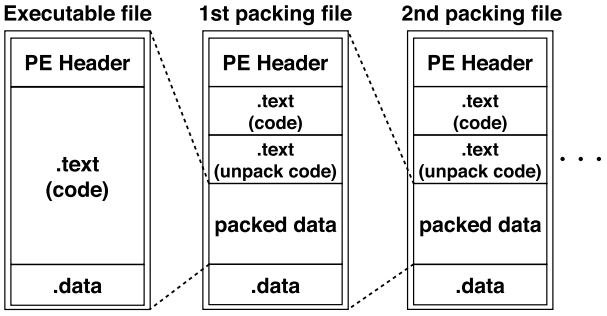


**Fig. 1** Process of multi-layer packing.

$c = p_n(\ldots(p_1(m))\ldots)$.

Figure 1 shows the process of multi-layer packing.

### 2.2 Entropy

In this letter, we consider the following entropy $H(\boldsymbol{X})$ for a byte-sequence $\boldsymbol{X}$ for the each component $X_k \in \mathbb{Z}_{2^8}$:

$$H(\boldsymbol{X}) = -\sum_{i=0}^{2^8-1} p(i)\log_2 p(i), \tag{1}$$

where $i$ denotes the value of 8-bit data ($i \in \mathbb{Z}_{2^8}$) and $p(i)$ denotes the probability of occurrence of $i$ in $\boldsymbol{X}$. Thus, the entropy $H(\boldsymbol{X})$ takes on a value of $H(\boldsymbol{X}) \in [0, 8)$.

### 2.3 $k$-Nearest Neighbor Algorithm

The $k$-nearest neighbor algorithm is a traditional pattern classification algorithm in the machine learning algorithm. In $k$-nearest neighbor algorithm, the Minkowski distance is used as index. The Minkowski distance $d$ between two points $\boldsymbol{x} = (x_1, \ldots, x_n), \boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbb{R}^n$ is defined as $d = \left(\sum_{i=1}^n |x_i - y_i|^p\right)^{1/p}$, where order $p$ is a real number of $p \geq 1$. Minkowski distance of $p = 2$ is very typical, which is called as Euclidean distance. We shall briefly describe the $k$-nearest neighbor algorithm in Algorithm 1.

### 2.4 Symbolic Aggregate Approximation (SAX)

Symbolic Aggregate approXimation (SAX) is a method of converting time-series data into string data.

SAX converts a time series of length $n$, $\boldsymbol{C} = (C_1, C_2, \ldots, C_n)$, into the sequence of length $w$, $\overline{\boldsymbol{C}} = (\overline{C_1}, \ldots, \overline{C_w})$, as follows:

Step 1: Transform time series $\boldsymbol{C}$ into normalized time series $\boldsymbol{C}'$ with mean of 0 and standard deviation of 1.

Step 2: The time series is divided into $w$ segments as

$$\overline{C_i} = \frac{w}{n}\sum_{j=\frac{w}{n}(i-1)+1}^{\frac{w}{n}i} c_j, \tag{2}$$

where $c_j$ is one point of time series $\boldsymbol{C}$ for all $j$.

**Algorithm 1** $k$-Nearest Neighbor Algorithm

**Input:** Hyperparameter: $k$, Sample vector: $\boldsymbol{x}$, Set of all sample vectors $S$.
**Output:** Class label: $L$

*// Calculate Minkowski Distances for all Samples*
**for** $\boldsymbol{s}$ in $S$ **do**
  Calculate Minkowski distance $\boldsymbol{s}$ and $\boldsymbol{x}$ as `d[`$\boldsymbol{s}$`]`.
**end for**
*// Choose $k$-Nearest Neighbor of Sample $\boldsymbol{x}$*
Sort `d[`$\boldsymbol{s}$`]` for all $\boldsymbol{s} \in S$ in ascending order.
Let the set $T$ be first $k$-number items of sorted vectors.
*// Determine the Class Label of Sample $\boldsymbol{x}$*
`n[`$l$`]`=0 for all class labels $l$.
**for** $\boldsymbol{t}$ in $T$ **do**
  Find the class label $l$ of $\boldsymbol{t}$.
  Increment the number of $l$: `n[`$l$`]`++.
**end for**
Determine the class label $L = l$ if `n[`$l$`]` is maximum for all labels $l$, by performing a majority rule.
**return** Class label $L$

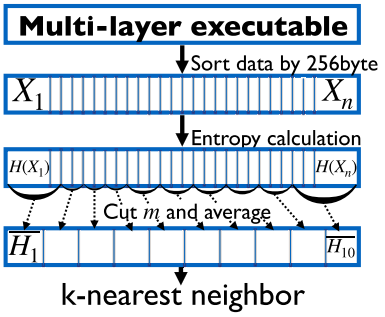


**Fig. 2** Proposed method.

## 3. Proposed Method

In this section, we shall propose an identifying method of the packer from multi-layer packed malware with $k$-nearest neighbor algorithm for entropies of the divided sequences. The Fig. 2 illustrates the analyzing process of the proposed method.

Step 1: The malware samples are divided into 256-byte chunks with the method proposed by Lyda et al [2]. Let $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)$ be a series of length $n$, s.t. the number of each element is 256-byte units.

Step 2: Calculate the entropy $H(\boldsymbol{X}_i)$ of each series $\boldsymbol{X}_i$ and Let the entropy series $h_i = H(\boldsymbol{X}_i)(i = 1, \ldots, n)$.

Step 3: Divide the entropy series $(h_1, \ldots, h_n)$ into $m$ sub-sequences as equal in length as possible. Let the average entropy of a sub-sequence be $\overline{H_k}(k = 1, \ldots, m)$.

Step 4: Let be the average entropy treated as the $m$-dimensional vector. We attempt to detect the packer of multi-layer packing by using the $k$-nearest neighbor algorithm method for the $m$-dimensional average entropy vectors.

**Table 1** Executable combination.

| 2nd packer \ 1st packer | Enigma | PElock | Themida | UPX | VMP |
|---|---|---|---|---|---|
| **Enigma** | No | Yes | No | Yes | No |
| **PElock** | No | Yes | Yes | Yes | Yes |
| **Themida** | No | Yes | Yes | Yes | No |
| **UPX** | No | Yes | No | No | No |
| **VMP** | No | No | Yes | No | Yes |

## 4. Experiment

Let the malware samples be 22 pseudo-malwares which are obtained from eagle0wl's crackme VOL.01 and VOL.02. Let the packer be the followings: Enigma [6], PElock [7], Themida [8], UPX [9] and VMP [10].

**Enigma Protector (Enigma)** The Enigma Protector is a packer which can be freely downloaded from enigmaprotector.com [6]. The packing algorithm is private.

**Demo version of PElock (PElock)** The demo version of PElock is a packer which can be freely downloaded from pelock.com [7]. The packing algorithm is private.

**Demo version of Themida x32 (Themida)** The demo version of Themida x32 is a packer which is provided by Oreans Technologies [8]. The algorithm is private.

**Ultimate Packer foe eXecutables (UPX)** The ultimate packer for executables is a packer provided by Free/Libre and Open Source Software [9]. UPX supports various executable-file formats of most OSs.

**Demo version of VMProtect Ultimate (VMP)** The demo version of VMProtect Ultimate is a packer provided in pelock.com [10]. The algorithm is private.

We show the first packer and the second packer in Table 1. The 'Yes-No' represents the answer if the packed malware is correctly executed.

Only 253 malware samples are correctly executed among the created $22 \times 12 = 264$ double-packed malwares by the combination of the status 'Yes'.

If the dimension $m$ is made larger, the feature amount will increase but the average entropies become unstable. Conversely, if $m$ is made smaller, the average entropy will be stable but the feature amount will decrease. Since the classification accuracy decreases in both cases, it can be said that there exists an optimum value of $m$. In this experiment, let the dimension of the sub-sequence be $m = 10$, which was determined by trial and error. We used $k$-nearest neighbor algorithm of the library of Scikit-learn with leave-one-out cross-validation.

The Fig. 3 shows the rate of correctly identifying the combinations of the packers for $k = 1, 2, \ldots, n$. The packer detection rate is about 49.6% at maximum when $k = 1$. From Fig. 3, we can conclude that the hyperparameter $k = 1$ is optimal in this experiment.

## 5. Conclusion

In this letter, we have proposed a method of specifying the packers from multi-packed executables with $k$-nearest neighbor algorithm of the entropies. By computer experiment, we can identify the packers about a half of malwares for double-packed malwares when $k = 1$. This result suggests that $m$-dimensional average entropy vectors of the proposed method represent good feature amount for identifying multi-layer packers. In order to confirm this, it is a future task to experiment by other classification methods.

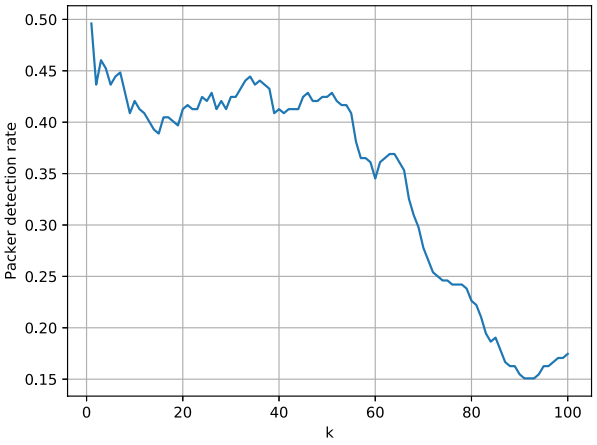


**Fig. 3** Rate of detecting packer.

For deeper analysis, it is future tasks to investigate the optimum value of $m$ and visualize the entropy vectors. Experimenting for three or more layers of multi-pack malware is also a future task.

## Acknowledgments

We would like to thank reviewers and editors for useful comments and suggestions.

### References

[1] R. Isawa, M. Morii, and D. Inoue, "An unpacking method based on instruction-trace similarity," Proc. CSS2016, 2016.
[2] R. Lyda and J. Hamrock, "Using entropy analysis to find encrypted and packed malware," IEEE Secur. Privacy Mag., vol.5, no.2, pp.40–45, 2007.
[3] L. Sun, S. Versteeg, S. Boztas, and T. Yann, "Pattern recognition techniques for the classification of malware packers," ACISP 2010, LNCS 6168, pp.370–390, 2010.
[4] M. Bat-Erdene, T. Kim, H. Park, and Heejo Lee, "Packer detection for multi-layer executables using entropy analysis," Entropy, vol.19, no.3, 125, 2017.
[5] "Web site of eagle0wl," http://www.mysys.org/eagle0wl
[6] "The Enigma Protector," https://enigmaprotector.com/en/about.html
[7] "PELock Software Protection & Software License Key System," https://www.pelock.com/products/pelock
[8] "Themida Overview," https://www.oreans.com/Themida.php
[9] "UPX," https://upx.github.io
[10] "Five Reasons To Use VMProtect," https://vmpsoft.com