Computer Security Symposium 2017
23 - 25 October 2017

Structural Entropy を用いたマルウェアの類似性評価についての一考察

東　結香[†1]

概要： 数多くの未知のマルウェアが日々確認されているが、機能や構造が類似したものも多く存在している。しかし、解析をしなければ類似性の有無を確認することは難しい。ハッシュが異なっても同じ挙動を取る可能性が高いものをまとめることができれば、コストのかかる処理(静的・動的解析)を減らすことが可能である。本論文では、Structural Entropy を用いたマルウェアの類似性評価方法を提案し、実際の検体を用いた分類結果を報告する。

キーワード：マルウェア，Structural Entropy，可視化、分類

A study for malware similarity evaluation method by Structural Entropy

Yuka Higashi[†1]

Abstract: Unknown malware are emerging every day. Their functions and structures are similar. However, it is difficult to confirm the presence or absence of similarity without analysis. If you know what is similar even if the hashes are different, it is possible to reduce costly processing (static and dynamic analysis).
In this paper, we propose a similarity assessment method of malware using Structural Entropy and report classification results using actual samples.

Keywords: Malware, Structural Entropy, Visualization, Classification

1. はじめに

昨今、サイバー攻撃による脅威は増大し続けている。攻撃のツールとして利用されるマルウェアも増加の一途をたどっている。2017 年第 1 四半期において、1 秒に 4 件以上の未知のマルウェアが確認されている。[1]しかし、ハッシュ値が異なるだけで、検体の機能および構造がほぼ同一のものも多く見受けられる。機能や構造が同一のものをグルーピングすることにより、確認すべき検体を少しでも減らすことができれば、同一ファミリー内で機能の変化や追加、新たな手法を用いたマルウェアを効率的に確認することが可能となる。しかし、機能や構造が同一であるかを判断するためには何らかの解析が必要でありコストがかかる。
Yara ルール[2]を用いて同一ファミリーを見つける手法や様々な特徴をもとに機械学習を用いた分類などが提案されている。Yara ではルールに合致するものを見つけるので、その性能は作成されるルールに依存する。ターゲットを定めその対象を収集する場合、Yara ルールは非常に有用であるがあるもののすべてのルールを作成することはコストが大きい。機械学習を用いた分類では多量の検体に対応することが可能だが、分類理由を確認することは難しい。
また、機械学習を用いた手法の性能は、アルゴリズムのみで決定するわけではない。特徴抽出やデータの質にも大きく影響を受ける。抽出した特徴によって類似性は異なる。仮に特徴をうまく抽出できたとしても、教師データやアルゴリズムによって、求めているアウトプットと異なることも生じる。さらに、多くのサンプルが必要となり、指標となる正解ラベルの付与も求められる。同一の機能・構造のものをまとめるという観点では、根拠が不明瞭でコストが大きい。

そこで、本研究ではハッシュが異なるが同一または類似している検体の分類を可視性の高い特徴を用いて行う。
具体的には、Structural entropy を用いたマルウェアの可視化及び類似性の調査を行い、尺度としての利用について考察を与える。

本論文の主な貢献は以下のとおりである。

- Structural entropy によるマルウェアにおけるデータ分布の可視化について提案した
- ハッシュは異なるが、検体自体ほぼ同一のものをグルーピングする際の閾値の調査

†1 トレンドマイクロ株式会社
Trend Micro Incorporated

2. 提案手法

本章では Structural Entropy を用いたマルウェアの可視化及び分類手法について述べる。本論文におけるマルウェアは Windows 実行ファイル形式のものを対象とし、類似性の比較対象はマルウェアのバイナリデータとする。
目的を達成するためには以下の要件を満たす必要があると考える。

(1) **人間にとって理解しやすい特徴**

(2) **特徴抽出の加工による影響が少ない**

特徴を抽出する際ための処理の影響によりデータが変化することがある。例えば、マルウェアの分類を行うときの特徴として逆アセンブルしたコードを用いる手法がある。しかし、現行のパーソナルコンピュータで使用される命令セットは可変長の IA-32 系列が主流であり、逆アセンブルが失敗することがある。その場合正しい特徴が抽出できず、結果に影響が生じる。

(3) **高速な処理**

多量の検体への適用するためには特徴抽出や類似度算出が短時間できることが望ましい。
本論文では上記の要件を満たす以下の手法を提案する。

Figure 1　提案手法

2.1 Structural Entropy

本論文におけるエントロピーH(n)とは平均情報量を差し、以下の式で定義されるものとする。

$$\mathrm{H(n)} = -\sum_{i=1}^{k} p_i(n) log_2 p_i(n)$$

Lyda[3]らの研究にあるよう、ファイルに含まれているデータによってエントロピーの値は異なる。プログラムの場合、セクションによって格納されているデータが異なり、Ivan[4]の研究で利用された Structural Entropy の概念を用いる。
バイナリデータに出現する値は 16 進の 0-FF であるため、k=256 とした。256 バイトを 1 ブロックとして重複させずに分割し、それぞれのエントロピーH(n)を算出する。算出した類似度を並べ特徴ベクトルxとする。

$$x = [H(1), \cdots, H(n)]$$

ただし、n= $\left\lfloor \frac{File\ \ size}{256} \right\rfloor$

2.2 可視化

本論文では、各ブロックのエントロピーをグラフとしてマッピングすることにより可視化を実現する。Python ライブラリの Metaplot を用い画像として出力した。サイズは 900 ピクセル×600 ピクセルとした。
マルウェアのサイズは様々であるため、ベクトルとして扱う際に何らかの処理が必要である。Ebringer[5]らは情報を落とすことによりベクトル長を合わせた。画像化した Strutual Entropy を特徴として用いることは、人間にとってわかりやすい特徴であるとともに、特徴を捨てることなく利用可能である。

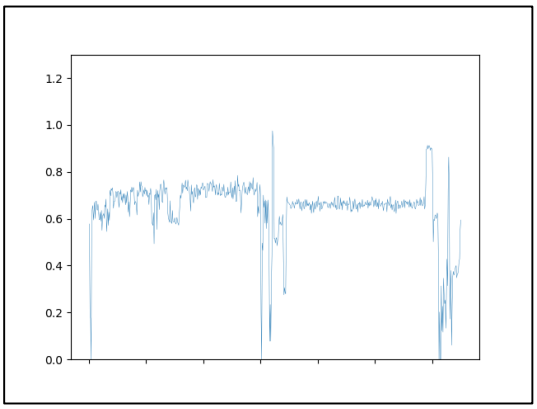


Figure 2　画像化した Structural Entropy

2.3 画像の類似度

視覚的に類似が見えたとしても、大量のデータを扱うためにはグループごとの精査が必要となる。２枚の画像の比較は目視でも可能だが、数千～数万枚のグラフの類似性を人の目で比較することは困難である。そこで、出力した画像の類似性を数値化することにより、グルーピングを行う。画像の類似性比較手法は複数存在する。本論文ではハッシュ値の算出および類似性の比較を高速に行うことが可能である ImageHash[6]を用いた。Perceptual Hash ともばれるこの手法はハッシュ関数の 1 種である。画像のサイズを縮小（デフォルトは 8×8 ピクセル）しそれぞれのピクセルに対し処理を行い 0 また 1 を出力する。そのビット列をハッシュ値として利用する。画像の縮小サイズによりハッシュの長さが決まる。類似性はハッシュ値のハミング距離を非類似度として用い比較可能である。本論文では以下の４つを適用し比較を行う。

Table 1　ImageHash 一覧

	算出方法
aHash	ピクセルがピクセル全体の平均値と比較する。平均以上の場合は 1、そうでない場合は 0 を出力する。
pHash	離散コサイン変換を用い算出した値を対象に全体の平均値と比較し 1 または 0 を出力する。
dHash	隣接するピクセルの差分をもとに算出する。
wHash	pHash と類似の手法。離散コサイン変換ではなく離散ウェーブレット変換を用いる。

3. 実験および考察

提案手法の性能を調査するために、独自に収集したサンプルを用いた。

3.1 データセット

2016 年 3 月 8 日～2017 年 8 月 6 日に独自に収集した 2795 2758 サンプル、および独自の自動アンパックツールを用いてアンパックを行ったもの 2103 サンプルを使用した。アンパック後のペイロードが同一のものが多く含まれており、アンパック後のユニークハッシュ数は 635 であった。アンパック後のサンプルは日本での検出が多いものに関しては Yara ルールによるラベルを付与した。Yara ルールがないもの（156/635）についてはトレンドマイクロ社の検出名を付与した。

Table 2 データセット

	サンプル数
総収集サンプル	2795
破損ファイル除去後	2758(破損: 37)
自動アンパック	2103
アンパック後のユニークハッシュ	635

Table 3 Yara ルールによるラベル(Top5)

ファミリー	ハッシュ数 ユニーク	マッピングの色
Cerber	104	青
Locky	75	紺
Urlsnif	65	水色
CrypMIC	33	シアン
GodzillaLoader	29	黄

3.2 画像の類似性と検体の類似性

提案は、データの分布が似ている検体は類似しているという仮定の上に成り立っている。本節では、画像が似ているものと実際のプログラムを比較する。

Figure 3　はアンパックした CryMIC Version3 の Structural Entropy を可視化したグラフである。

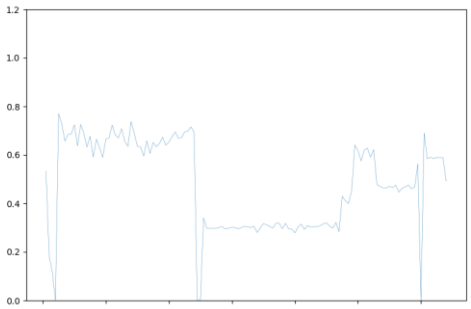


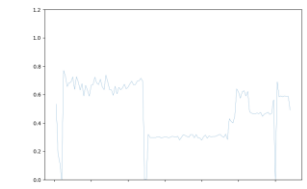
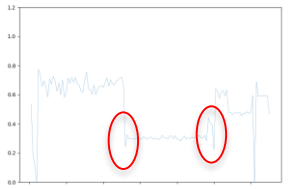
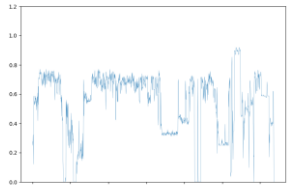
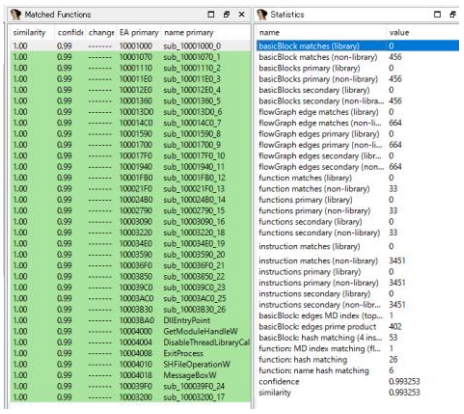
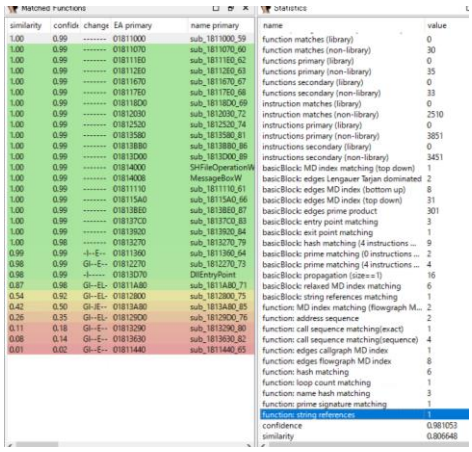
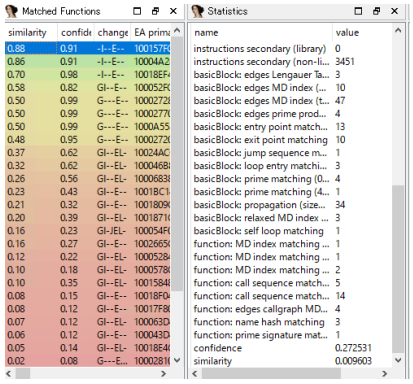
Figure 4　CrypMIC version3 のグラフ

この検体と 3 種類の検体を比較する。これらはすべてアンパック後の検体である。

(1) ハッシュが異なる CrypMIC version3
(2) CrypMIC version2
(3) Crypxxx version　4 系

Table 4 に Structural Entropy を可視化したグラフおよび Bindiff[7]での Confidence および Similarity をまとめた。CrypMIC と CrypXXX ではグラフが大きく異なる。Bindiff の結果でも近しい関数はほぼなく、Confidence も Similarity 低い。CrypMIC 同士でもバージョンが異なることにより、一部関数が異なり、同じバージョンのものと比べると Confidence も Similarity 低くなっている。グラフにおいてはかなり近い形状ではあるものの、2 か所ほどエントロピーの値が大きく異なっている。

Table 5　比較対象のファイル

CrypMIC version 3	CrypMIC version 2	Crypxxx version　4 系
		
		

上記の例だけでなく、他のサンプルでも Structural Entropy を可視化したもの類似性と検体の類似性の関係を確認するために、アンパック後の画像の非類似度を用い 2 次元にプロットした。具体的には、ImaseHash のハッシュの長さと取得するハッシュ関数を変化させたデータを多次元尺度構成法によって可視化を行った。Yara ルールによるラベル Top5 に着色を行い、類似性を持つかどうか確認した。比較を行ったパラメータは以下のとおりである。

- ハッシュ関数：aHash, pHash, dHash wHash
- ハッシュ長：16, 32, 64

結果、whash はどのハッシュ長であっても類似性を表現することが難しいことが分かった。今回のデータでは pHash による非類似度の分布が最も適切だと判断し、本論文では以後 pHash を用いる。

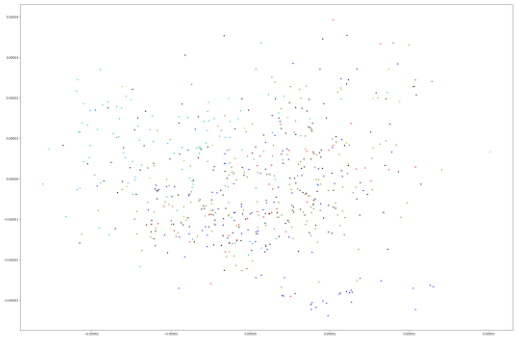


Figure 5 ハッシュ長 32 の wHash

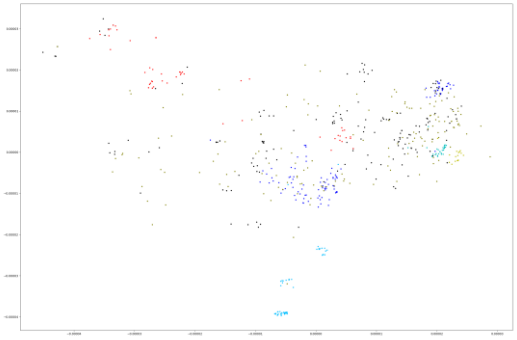


Figure 6 ハッシュ長 32 の pHash

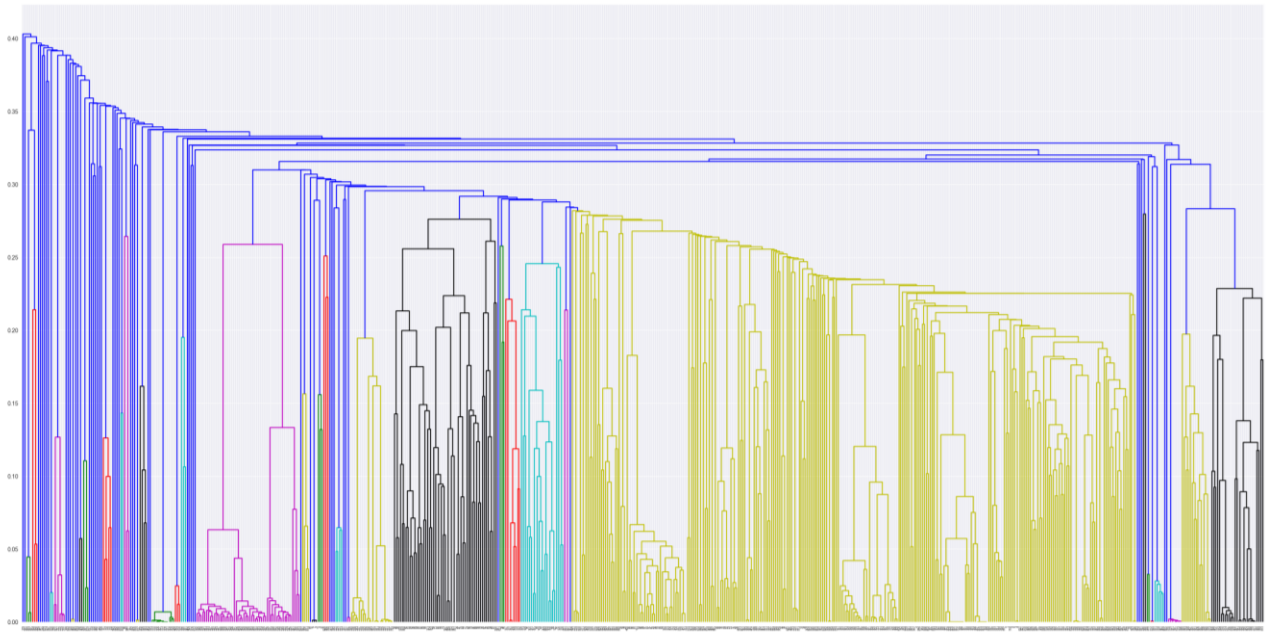


Figure 7　アンパック後の検体のデントログラム

3.3　アンパック後の検体の分類と考察

非類似度を用い、ハッシュ長 64 の phash を average 法にて階層型クラスタリングを行った。切断した高さごとのクラスタ数は以下のとおりである。

Table 6　クラスタの概要

高さ	クラスタ数	2 つ以上のハッシュが含まれるクラスタ
0.15	247	87
0.1,	317	94
0.05	394	78
0.03	435	65

(1)　**0.03 で切断**

クラスタは 435 生成された。ほとんどのクラスタは 1 つのハッシュのみで構成されていた。2 つ以上のハッシュが含まれるクラスタにおいて、ラベルがついているものはすべて同一ファミリが分類されていた。また、同一クラスタには同時期に配られたものやランサムウェア Locky では暗号化後のファイル拡張子ごとに別クラスタに分類ができていた。

(2)　**0.05 で切断**

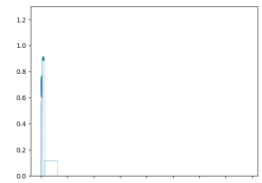
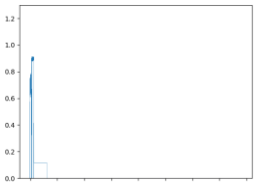
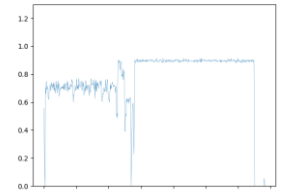
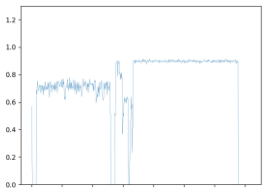
クラスタ 394 生成された。CERBER は一部バージョンが異なるものが同一グループになっていたが、そのほかのファミリーは、配布日やバージョンが異なるものは異なるグループに分類されていた

CERBER のバージョン違いが含まれてしまう原因としては padding が考えられる。サンプルを確認したところ、約 9,2Mbyte は”0”で埋め尽くされていた。パッカーとアンパック後の検体にも関連性が見られた。そのため、padding 自体一つの特徴と捉えることもできるため、一概に誤った分類とは言えない。

(3)　**0.1 および 0.15 で切断**

配布時期やバージョンが異なるサンプルに関しては一部のクラスタで混じってしまったが、ファミリーとして誤った分類は発生しなかった。

Table 7　バージョンが異なる CERBER の比較

CERBER version2	CERBER version4
	
	

(4)　**クラスタに所属しなかったサンプル**

0.15 で切断したとしても、161 検体は自身のみのクラスタになっていた。原因の 1 つは先に述べた通り、padding に

よって、グラフの形はそのままに縮小してしまったものである。ImageHash は同じ場所のピクセルを比較するため、グラフが縮小してしまい、視覚的には類似性が見えるが、数値としては異なってしまう。一定以上エントロピーの値が”0”出会った場合、削除することにより改善する可能性がある。また、切断場所によって漏れてしまったものもある。クラスタ ID や非類似度行列を確認したところ、もう少し上部で切断することによりグルーピングできることが分かった。

今回のデータセットにおいては、配布時期やファミリーを含めたグルーピングを行う場合は 0.05 以下での切断、あれば可能であることが分かった。一方、ファミリーでの分類でよい場合はもう切断場所の検討が必要であることが分かった。

3.4 パッカーとアンパック後の検体の比較

3.4 節と同様に、非類似度を用い、ハッシュ長 64 の pHash を average 法にて階層型クラスタリングを行った。

高さ	クラスタ数	2 つ以上のハッシュが含まれるクラスタ
0.3	631	242
0.15	1792	347
0.1,	2105	301
0.05	2341	208
0.03	2416	171

アンパック後の検体が Yara ルールによってラベリングされているものを対象に比較を行う。高さ 0.03 および 0.05 で切断する場合、配布時期やバージョンの誤りもなく、グルーピング可能であった。高さ 0.1 での切断を行う場合、CrypMIC において 1 検体のみバージョンが異なるクラスタに分類されていた。それ以外については、配布時期やバージョンの誤りもなく分類可能であった。一方、0.15 で切断を行う場合、同一クラスタに CERBER および Zbot が分類された。グラフを確認したところ非常に類似していた。コードを確認したところどちらも NSIS（Nullsoft Scriptable Install System）により圧縮されたものであった。

4. 今後の課題

ラベルに沿ったグルーピングができたものも存在したが、切断を行う値によっては、人の目で見た際に類似しているものでも同一グループに分類することが難しいものが存在した。また、pHash を特徴とした場合、グラフの形が類似していても描画される部分のずれの影響を受けてしまうため、目的に応じて padding の処理の検討を行う必要がある。また、今回 Yara ルールによるラベリングを用いて類似性の検討を行ったが、Yara ルールによってすべてが網羅されているわけではない。SandBox のログ等ほかの観点での類似性と合わせて分析を行うことにより、Structural Entropy と機能や構造の類似性について考察を行うことができると考える。

5. 関連研究

エントロピーを用いたアプローチについて述べる。特徴としてエントロピーを用いた研究はパッカーの推定に関するものが多い。Lyda,[3]は 256byte ごとにエントロピーを算出し、平均値や最大値を用いてファイルの状態を推定した。テキストファイル、通常の実行ファイル、パックされた実行ファイル、暗号化された実行ファイルの 4 種をエントロピーから判断した。

Perdisci[8]らは、セクションや API の数に加えセクションごとのエントロピーの値を用いて、パックされたマルウェアであるかどうかを推定する手法を提案した。Ebringer[5]らはファイル全体で構成したハフマンツリーを用い、エントロピー推定を行った。エントロピーを推定する区間や特徴として使用するエントロピーの選定、対象のファイルサイズなどの条件を変化させ、提案手法を用いたパッカーの推定の適切なパラメータを明らかにした。Ivan[4]は 256byte を 1 ブロックとして、1 byte ずつずらしたエントロピー（Structural Entropy）を用いた。ポリモーフィックマルウェアにも対応できるよう、ウェーブレット解析を用いセグメントを推定し、セグメントごとの Structural Entropy の編集距離を用いたマルウェアの比較手法を提案した。

6. おわりに

本論文では Structural Entropy を可視化した画像を用いマルウェアの分類および類似性評価を行った。配布時期やバージョンが同じものが同一クラスタに分類されることを確認した。一方で必要な類似性の定義や、部分的に一致するような特徴に対して、不十分な部分があることが明らかになった。今後はほかの特徴等も利用し改良を行っていく。

謝辞 本論文を執筆するにあたり、実験に協力いただいたトレンドマイクロ株式会社川畑公平氏、中谷吉宏氏に深く感謝する。

参考文献

[1] “脅威レポート”. https://www.mcafee.com/jp/resources/misc/infographic-threats-report-jun-2017.pdf, (参照 2017-08-20).
[2] “YARA - The pattern matching swiss knife for malware researchers “, http://virustotal.github.io/yara/, (参照 2017-08-20).
[3] Lyda, Robert, and James Hamrock. "Using entropy analysis to find

encrypted and packed malware." IEEE Security & Privacy 5.2 (2007).
[4] Sorokin, Ivan. "Comparing files using structural entropy." Journal in computer virology 7.4 (2011): 259-265.
[5] Ebringer, Tim, Li Sun, and Serdar Boztas. "A fast randomness test that preserves local detail." Virus Bulletin 2008. Virus Bulletin Ltd, 2008.
[6] “GitHub - JohannesBuchner/imagehash: A Python Perceptual Image Hashing Module” https://github.com/JohannesBuchner/imagehash(参照 2017-08-27).
[7] “zynamics.com – BinDiff” https://www.zynamics.com/bindiff.html(参照 2017-08-27).
[8] Perdisci, Roberto, Andrea Lanzi, and Wenke Lee. "Classification of packed executables for accurate computer virus detection." Pattern recognition letters 29.14 (2008): 1941-1946.