利用者名：コンピュータサイエンス学部　准教授　宇田 隆哉

Scopus

Title: Investigation of The Latest Malware Detection Engines and Lightweight Byte n-gram Methods with Real Custom Malware

（最新のマルウェア検出エンジンの調査と実際のカスタムマルウェアを用いた軽量なバイト n-gram 手法）

Authors: Ryuya Uda, Shinnosuke Araki

（荒木 真之介（東京工科大学　卒業生）、宇田 隆哉（東京工科大学　准教授））

Journal: Proc. 16th International Conference on Computer and Automation Engineering, 2024.

掲載年月：2024 年 7 月

研究概要：最新のマルウェア検出エンジンが、どのようにマルウェアを検出しているのか、実際のマルウェアを改変して確認してみました。そして、改変したマルウェアを、軽量なバイト n-gram 手法で高精度に検出できることを示したかったのですが、2～5 検体しか検出できない結果になってしまいました。

研究背景：皆さんのパソコンにもウイルス対策ソフトウェアがインストールされているものと思います。このウイルス対策ソフトウェアが対象とするのは、一般的にコンピュータウイルスと呼ばれているものですが、ファイル単体で動作するものは正確にはこの定義に当てはまりませんので、悪意のあるソフトウェアを総称してマルウェアと呼ぶようになりました。これらのウイルス対策ソフトウェアにはマルウェア検出エンジンが含まれており、あるプログラムがマルウェアかそうでないか見分けています。これさえインストールされていれば安心と思う人もいるかと思いますが、それぞれのマルウェア検出エンジンがどのようにしてマルウェアを検出するのかについては、詳細な部分が公開されておらず、利用者はよくわからないまま使っているというのが現実です。そこで、すでにマルウェアと判定されているファイルを少し改変し、既存のマルウェア検出エンジンに入力してみたら反応がどのように変わるのか、本研究で調べてみました。

研究成果：すでにマルウェアと判定されているファイルを集めました。あまり古いマルウェアですと、現在の状況にそぐわない可能性がありますので、VirusShare.com から 2017 年より新しい「Win32」のものだけを 162 個ダウンロードしました。マルウェア検出エンジンとしては、Virus Total を使いました。マルウェアを改変する方法としては、次の 3 通りを試しました。(1)One Byte Addition：ファイルの最後に「FF」の値を持つ 1 バイトを付け加えます。(2)Many Bytes Addition：ファイルの最後に“Google”という文字列を繰り返してたくさん付け加えます。(3)Random Bytes Insertion：ランダムな 4 バイトをマルウェアの途中に挿入します。なぜ(2)の文字列が“Google”なのかといいますと、Arai らの研究で同様のことを行っていたからです。Google は誰もが知っている有名な企業で、その文字列を大量にマルウェアに付け加えると、マルウェア検出エンジンがそれだけで良性と判定するということだそうです。(3)は、論文の図 1 のように行っています。マルウェアの 4 バイト区切りの位置に、各箇所 50%の確率で、ランダムな 4 バイトを挿入します。プログラムとして今まで通り動作するようにするのであれば、参照するアドレスがずれますので、それを調整しないといけないのですが、ほとんどのマルウェア検出エンジンが部分的なパターンマッチングを行っているだけと考え、それについては考慮しませんでした。さらに厳密にいえば、プログラムとして動かせる命令がある領域に挿入する場合、動作に影響がないような 4 バイトをよく考えて挿入しなければなりませんので、逆アセンブルが必要になります。One Byte Addition と Many Bytes Addition は似たような結果になりました。例えば、「3da3」のファイルでは、元は 71 個中 56 個のエンジンがマルウェアと判定していますが、One Byte Addition では 72 個中 50 個、Many Bytes Addition では 72 個中 49 個になりました。まず、たった 1 バイト末尾に加えただけでマルウェアを検出しなくなるエンジンは、シグネチャによるパターンマッチングを使っています。約 6 個のエンジンがこの古いタイプの検出方法を使っていることになります。One Byte Addition と Many Bytes Addition で差が 1 個程度であることは、“Google”という有名企業の文字列を加えたことが原因ではなく、1 バイトでも加えればシグネチャによるパター

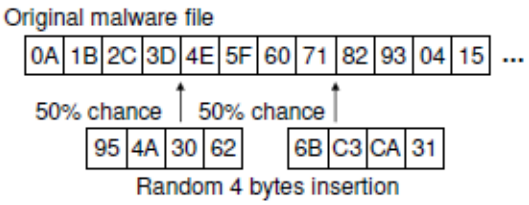

Fig. 1. Method of random bytes insertion.

TABLE II
RESULTS OF RANDOM BYTES INSERTION IN MALWARE DETECTION

Detected Files	0	1	2	3 or more
Number of Files	152	3	7	0

ンマッチングが行えなくなることによるものと思われます。Random Bytes Insertion の結果は、論文の表 2 のようになりました。163 個中 152 個のマルウェアのファイルが、すべてのエンジンに“マルウェアではない”と判定されました。今回の 4 バイトの挿入は厳密なやり方ではないので絶対的な評価はできませんが、これらのエンジンの中に、部分的なパターンマッチングだけを行っているものがあるのだとしたら、攻撃者に検出を簡単に回避されてしまう危険性があります。

社会的・学術的なポイント：従来のマルウェア検出エンジンでは、少し改変しただけのマルウェアが検出できなくなる一方、軽量なバイト n-gram 手法では高精度で検出できることを示したかったのですが、そうはなりませんでした。これは、本研究では、2017 年より新しいマルウェアのみを検体として使ったことが理由と考えられます。詳細の説明は省略しますが、Windows の亜種マルウェアが極端に少なくなり、上位 500 個程度の特徴では機械学習に十分ではなくなったということです。先述の上位 500 個の特徴の「500」の部分の数が増えれば、おそらく軽量なバイト n-gram の手法も有効なのですが、そうしますと処理が軽量ではなくなってしまいます。本研究室では、Kita らの研究で、このバイト n-gram を高速に処理可能な方法を見つけたのですが、それは亜種マルウェアにしか適用できません。実は、その後に研究が進み、亜種マルウェアでなくともバイト n-gram を高速に処理可能な方法を見つけました。2024 年 3 月の情報処理学会コンピュータセキュリティ研究会で発表していますので、そちらを是非ごらんください。

用語解説：

Win32：32bit 版の Windows で動作するということ。64bit 版の Windows では動作しないということではなく、32bit 版の命令を使っているということを指します。

FF：16 進数表記で FF です。F は 10 進ですと 15 で、2 進数ですと 1111 となります。1 バイトは 8bit ですので、16 進数表記の FF は 2 進数で 11111111 の 8bit となります。

逆アセンブル：機械語をアセンブリ言語に戻すこと。

シグネチャ：ファイル全体のハッシュ値を記録しておくもの。つまり、このハッシュ値のファイルはマルウェアということになります。

5 分割交差検証：検体全体を 5 つのグループに分割し、4 つをトレーニングに、残りの 1 つをテストにして機械学習を行い、テストのグループにこの 5 つすべてが使われるように 5 回機械学習を行う方法のこと。

亜種マルウェア：同じ動作もしくは類似の動作をするが、シグネチャが異なるマルウェアのこと。