アンチウイルスによる誤検知の調査と対策の提案

岩本 一樹[1,a)]

概要：マルウェアへの感染を防ぐために，アンチウイルスソフトウェアが使われている．しかしアンチウイルスソフトウェアは，しばしば正常なソフトウェアを誤ってマルウェアとして検知すること（誤検知）がある．誤検知が発生した場合，誤検知されたソフトウェアはユーザからの信頼を失う可能性があり，また開発者は誤検知の対応に追われることになる．一方，ユーザもコンピュータがマルウェアに感染したと誤った判断することで不要な手間をかけたり，ソフトウェアが使えなくなることで生産性を低下させる可能性がある．さらに，誤検知が多発するならば，アンチウイルスソフトウェアはユーザからの信頼を失い，本当の検知が見過ごされる可能性がある．そこで，本研究では事例と実験から誤検知が発生する原因を探り，ソフトウェア開発者がとるべき対策を提案する．

キーワード：誤検知, アンチウイルス, マルウェア, コードサイニング証明書

Survey of False-Positive by Antivirus Software and Proposal of the Countermeasures

Kazuki Iwamoto[1,a)]

Abstract: Anti-virus software is used to prevent malware infection. However, antivirus software sometimes has the problem to detect benign software as malware (false-positive). If false-positive occurs, the detected software may lose the trust from the users, and the developer may be swamped with support for false positive. On the other hand, if the users decided incorrectly that the computer was infected with malware, they would take unnecessary task and/or their productivity would be reduced by losing the software. Moreover, if false-positive occurs frequently, anti-virus software would lose trust from the users, and true detection would be ignored. Therefore, in this paper, we explore the conditions under which false-positive occurs from instances and experiments, and we propose countermeasures to be taken by software developers.

Keywords: False-Positive, Anti-Virus, Malware, Code Signing Certificate

1. はじめに

著者はソフトウェアの開発・配布サイト「OSDN」で自作のソフトウェアを配布している．2018 年 6 月 25 日，同サイトで Virus Total の API を利用したマルウェアの検査が開始された [1]．2017 年以降に更新した，著者のソフトウェアのアーカイブやインストーラーの検査結果を**表 1** に示す．検査結果の書式は「検知数/検査したアンチウイルスエンジンの数」である．2018 年 6 月 25 日の時点で少なくとも 6，最も多い検知数は 33 であり過半数を超えるアンチウイルスエンジンで検知された．当然，これらは誤検知である．

著者らは MITB 攻撃型ウイルスの振る舞いで検知する PhishWall プレミアムの「PhishWall クライアント」[2] を配布している．同ソフトウェアの Firefox および Chrome 版インストーラー「pw-sb-5.1.exe」は 2018 年 5 月 29 日の時点で Virus Total の 1 つのアンチウイルスエンジンで検知され，2018 年 6 月 28 日の時点では 5 つのアンチウイルスエンジンで検知された．当然，これらは誤検知である．

ソフトバンクの秋山秀三氏は「SoftBank World 2018」の

[1] 株式会社セキュアブレイン
SecureBrain Corporation
[a)] kazuki_iwamoto@securebrain.co.jp

表 1　自作ソフトウェアの検査結果（2018 年 6 月 25 日）

ファイル名	検査結果
iwmutlis-0.0.7-win32.msi	6/60
iwmutlis-0.0.7-win32.zip	19/62
iwmutlis-0.0.8-win32.msi	15/61
iwmutlis-0.0.8-win32.zip	20/62
iwmutlis-0.0.9-win32.msi	20/61
iwmutlis-0.0.9-win32.zip	26/63
iwmutlis-0.1.0-win32.msi	23/60
iwmutlis-0.1.0-win32.zip	33/62
tm019xja.msi	17/60
tm019xja.zip	17/61

講演で，RPA（Robotic Process Automation）ツールで作成したソフトウェアロボットが EDR（Endpoint Detection and Response）で誤検知されることがあると報告した [3]．前述の Virus Total は実行前の誤検知であるのに対して，この事例では RPA が実際に動作している時にその動作を誤検知する．

ソフトウェアそのものの誤検知ではなく，ソフトウェアを配布する Web サイトに対する誤検知の問題であるが，2012 年に発生した「いじくるつくーる」の事例 [4] がある．この事例では何度もアンチウイルスベンダーと連絡をとり，解決までに 1 年 8 ヶ月を要した．おそらく日本国内では初めての誤検知の問題が広く一般に取り上げられた事例 [5] である．これに関連して，ソフトウェアを配布する Web サイトの窓の杜では，アンチウイルスソフトウェアの一部の機能については誤検知が多いため、その結果を参考程度と捉えるという扱いをしていることを公表した [6]．

一方，Virus Total は 2018 年 6 月 19 日に新たなサービス「VirusTotal Monitor」を発表した [7], [8]．このサービスでは，ソフトウェア開発者が公開前のファイルを提供してアンチウイルスソフトウェアで検査を行う．誤検知が発生した時にはアンチウイルスベンダーに報告され，公開前にアンチウイルスソフトウェアの対応を促す．これにより，公開後のソフトウェアがアンチウイルスソフトウェアに誤検知されることを防ぐ．また同発表では，誤検知によるソフトウェアの信用の低下やソフトウェア開発者の負担を指摘している．

以上のことから，

- アンチウイルスソフトウェアの誤検知は著者らだけの問題ではなく，広く一般的な問題である．
- 誤検知されたソフトウェアは信頼を失う可能性がある．
- 誤検知が多発すれば，アンチウイルスソフトウェアの警告が軽視される可能性がある．
- 誤検知の解決はソフトウェア開発者の負担が大きい．
- 誤検知によりユーザも不利益を被る．

と言える．そこで本研究では事例と実験から誤検知が発生する原因を探り，ソフトウェア開発者がとるべき対策を提案する．

本論文の構成は次のとおりである．2 章では関連研究について記述する．3 章では著者らの実際に起こった事と対策について記述する．4 章では本研究の目的を達成するために行った実験と結果を記述する．5 章では 3 章の事例と 4 章の実験を考察する．6 章では対策を提案する．最後に，今後の課題を 7 章に記述する．

2. 関連研究

アンチウイルスソフトウェアを評価する機関として，AV-TEST[9] や Virus Bulletin[10] がある．AV-TEST[9] では 15 年前から収集している正常なファイルのアーカイブのスキャンと，標準的なソフトウェアのインストール過程での誤検知を評価している．

Virus Bulletin[10] では 400,000 以上の正常なファイルから 25%以上が PE ファイルになるように約 100,000 のファイルをランダムに選び誤検知を評価している．VB100 の認定を得るためには誤検知率が 0.01%以下である必要がある．

しかし，これらの結果はアンチウイルスソフトウェアの改良にはつながるが，ソフトウェア作者に対して対策を提案するものではない．また，一般にはアンチウイルスソフトウェアの評価ではアンチウイルスソフトウェアの誤検知よりも検知率が注目されている．

Kim らの研究 [11] では，アンチウイルスソフトウェアがデジタル署名されたマルウェアを検知しない場合があることを指摘している．Kim らは不正な形式の期限切れの証明書のデジタル署名を 2 つ抽出した．それらを 5 種類の既知のマルウェアに付加することで 10 種類の検体を作成し，アンチウイルスソフトウェアで検査した．Kim らはこの誤ったデジタル署名でアンチウイルスソフトウェアの検知が回避できる場合があることを示した．Kim らの研究とは反対に，本研究では正規のコードサイニング証明書を用いてデジタル署名された正常なソフトウェアの誤検知について実験・考察する．

3. 事例

本章では著者らが配布する PhishWall クライアントが誤検知を解決する過程と，表 1 に示す著者のソフトウェアの誤検知数の変化を記述する．

3.1 PhishWall クライアント

PhishWall クライアントの Firefox および Chrome 版インストーラーの誤検知数の変化を**図 1** に示す．図 1 の赤はバージョン 5.1.29.246，青はバージョン 5.1.30.264 の誤検知数である．

3.1.1 アンチウイルスベンダーへの報告

著者らの主な対策はアンチウイルスベンダーに報告して

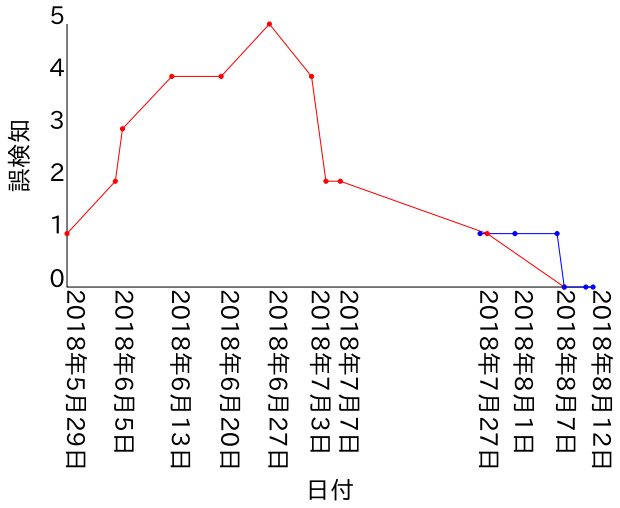

図 1　PhishWall クライアントの誤検知数

誤検知をなくすことである．本項にアンチウイルスベンダーへの報告の過程を記述する．誤検知した各アンチウイルスエンジンは EG1〜EG5 と表記する．2008 年 8 月 13 日の時点で誤検知は発生していない．

6 月 27 日　Virus Total に問い合わせる．

6 月 29 日　Virus Total からアンチウイルスベンダーに連絡してほしいという返答と連絡先の一覧を得る．

7 月 3 日　連絡を行う前に EG4 は Clean になったので，EG4 には何もしない．EG1，EG2，EG3 にメールで連絡，EG5 の Web フォームに登録する．

7 月 3 日　EG2 から次回更新で Clean になると回答を得る．

7 月 4 日　EG1 から Web フォームを案内される．EG1 の Web フォームに登録する．

7 月 6 日　EG1，EG2，EG3 が Clean となっていることを確認する．

7 月 6 日　EG4 で再び誤検知されていることを確認する．EG4 の Web フォームに登録する．

7 月 11 日　EG5 の Web フォームに再登録する．

7 月 27 日　EG5 が Clean となっていることを確認する．

7 月 27 日　8 月 1 日にバージョン 5.1.30.264 に更新予定なので，EG4 で誤検知された状態だがバージョン 5.1.29.246 については報告を終了する．

7 月 27 日　公開予定のバージョン 5.1.30.264 が EG1 で誤検知されるので，EG1 の Web フォームに登録する．

8 月 1 日　EG1 が Clean となっていることを確認する．

8 月 1 日　EG4 で誤検知されていることを確認する．EG4 にメールで連絡する．（Web フォームにアクセスできなかった）

8 月 2 日　EG4 の Web フォームに登録する．

8 月 7 日　EG4 にメールで解析終了の期日を問い合わせる．

8 月 13 日　EG4 が両バージョンで Clean となっていることを確認する．

3.1.2　その他の原因の推測と対策

著者らは Firefox および Chrome 版インストーラーの脆弱性 [12], [13] を修正した．著者らは，この修正が誤検知の原因の 1 つではないかと推測した．しかし本論文の執筆時点では，この原因と推測した修正に対する対策はしていない．

また修正されたバージョン 5.1.29.246 では，ファイル名を「setup.exe」から「pw-sb-5.1.exe」に変更した．著者らは，新しいファイル名が二重拡張子にように見えるため，誤検知されたと推測した．著者らはバージョン 5.1.30.264 でファイル名を「pw-sb-5_1.exe」に変更した．

3.1.1 項の報告の過程で，各アンチウイルスベンダーから誤検知の原因や誤検知を発生させないための情報などの提供はなかった．そのため，本項に記述する誤検知の原因の推測が正しかったか否かは不明である．

3.2　自作ソフトウェア

著者は OSDN でいくつかのソフトウェアを公開しているが，2017 年以降に更新したソフトウェアは「IWM Utilities」（以下「iwmutlis」）と「Text maid」（以下「tmaid」）の 2 種類である．

iwmutlis はバイナリ解析ユーティリティであり，主に著者のアンパックに関する研究 [14] のために作成したプログラムと，シェルコード特定の研究 [15] の元になったプログラムで構成される．iwmutlis のアーカイブ・インストーラーには 11 種類の実行可能ファイルが含まれる．11 種類のうち 8 種類のソースコードは特定の CPU や OS に依存しておらず，Windows 以外にも UNIX 系 OS などでコンパイルして実行することができる．残りの 3 種類は 32 ビット Windows 専用のプログラムである．

tmaid は 32 ビット Windows 専用のテキストエディタである．tmaid のアーカイブ・インストーラーには 1 つの実行可能ファイルが含まれる．tmaid は Windows の API だけを使用しており，第三者のライブラリやランタイムライブラリには依存しない．

これらの自作ソフトウェアの実行可能ファイルの誤検知の変化を**図 2** に示す．図 2 の線の色は，最大誤検知数が多いファイルは赤に近く，少ないファイルは青に近い．8 月以降に公開したファイルはコードサイニング証明書でデジタル署名されている．iwmutlis と tmaid のバージョンと公開日，デジタル署名の有無を**表 2** に示す．

3.2.1　原因の推測と対策

iwmutlis はバイナリ解析ユーティリティでありマルウェアを扱っている．しかし OS 汎用のプログラムはメモリ確保と演算，文字列操作，ファイル入出力で構成されており，Windows に固有の機能は使われていない．32 ビット Windows 専用のプログラムは Windows の API を利用す

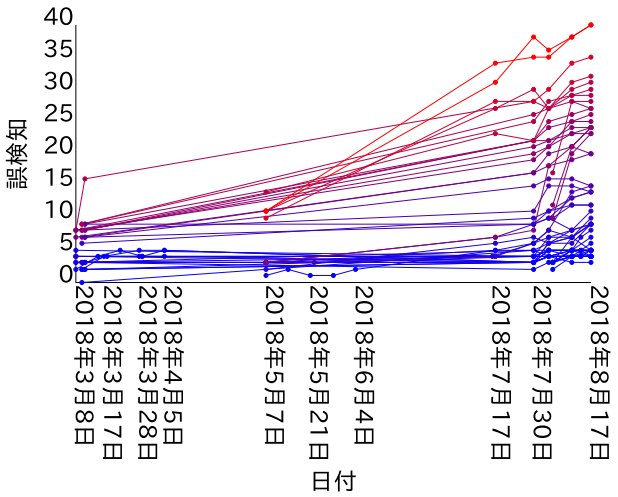

図 2　自作ソフトウェアの誤検知数

表 2　自作ソフトウェアの公開日，バージョン，デジタル署名

公開日	iwmutlis	tmaid	デジタル署名
2017 年 7 月 18 日	0.0.7		
2017 年 9 月 17 日	0.0.8		
2018 年 1 月 29 日	0.0.9		
2018 年 5 月 7 日	0.1.0	1.9X	
2018 年 7 月 16 日		1.9Y	
2018 年 7 月 30 日		1.9Z	
2018 年 8 月 5 日	0.1.1	3.0.0	✓

るが，プログラムは短く，誤検知を引き起こす理由は思いつかない．また tmaid は単純なテキストエディタであり，誤検知を引き起こす理由は思いつかない．

しかしこれらのプログラムに共通で，一般的ではない仕様は下記の 2 つがある．

- コンパイラが標準でリンクするランタイムライブラリを使用せず，自作のライブラリを使用し msvcrt.dll とリンクしている．本論文ではこの仕様を「自作ライブラリ」と称する．
- コンパイラで実行可能ファイルを作成後，実行可能ファイルに含まれる MS-DOS スタブやデバッグ情報などの実行に必要がないデータを削除し，可能ならばセクションの隙間を減らすことでファイルサイズを縮小する加工を行っている．本論文ではこの仕様を「再構築」と称する．

著者はこれらの仕様が一般的ではないため，誤検知されたと推測した．

本論文の執筆時点では，これらの仕様は変更していない．著者は自作ソフトウェアの誤検知に対しては，特に対策は行わなかった．

4. 実験

著者らは本章の実験で 3.2.1 項の自作ソフトウェアの誤検知の原因の推測を検証する．本論文ではコンパイラのオプション変更やファイルの改変などを「小細工」と称する．著者らは小細工により多様な実行可能ファイルを作成してアンチウイルスソフトウェアで検査することで，小細工が誤検知に及ぼす影響を調査した．各々の小細工は英数字のラベルで表す．本章の各節でラベルを示し小細工について説明する．

4.1 節と 4.2 節の小細工はコンパイラの設定だけで実現しており，特に 4.1 節の小細工は実際のソフトウェアの開発でもありうる設定の変更である．4.3 節の小細工は一般的とは言えないが，プログラムの機能上の必要性があって行う小細工である．4.4 節の小細工はコンパイラが作成したバイナリを加工しており，プログラムの機能上の必要性は乏しい．

3.2.1 項の推測の自作ライブラリは 4.3 節の小細工 il に相当し，再構築は 4.4 節の小細工 rb に相当する．表 2 に示す自作ソフトウェアは小細工「o2 md il rb」または「o2 md il rb ct」に相当する．

著者らは実験のために，11 種類のアンチウイルスソフトウェアを**表 3** に示す環境のゲスト OS にインストールした．アンチウイルスソフトウェアは AV01〜AV11 と表記する．誤検知の結果を**表 6** に示す．誤検知がなかった AV04，AV07，AV08，AV09，AV11 は表 6 から除く．

表 3　調査に用いた環境

ホスト OS	OS X 10.11.6
仮想環境	VMware Fusion 8.5.10
ゲスト OS	Windows 7 Professional SP1 32bit
コンパイラ	Microsoft C/C++ Optimizing Compiler Version 19.00.23918 for x86
リンカー	Microsoft Incremental Linker Version 14.00.23918.0

4.1 バージョンとコンパイラオプション

tmaid のバージョンとコンパイラのオプションを変更することで，同一の機能をもつ異なる実行可能ファイルを作成した．小細工のラベルと説明を下記に記述する．なお，小細工の組み合わせの中で現実的であると思われる 10 種類の実行可能ファイルを作成した．

参考までに，バージョン 1.9X でコンパイラのオプションを変更して作成した実行可能ファイルと小細工「19y o2 md」の実行可能ファイルの各ファイル間の類似度を**表 4** に示す．著者のマルウェアの分類の研究 [16] の類似度を，本研究の実行可能ファイルに適用して求めた．この研究では，類似度が 0.2 以上ならば，実行可能ファイルは関連性があると考えられる．表 4 の中の類似度のいくつかは 0.2 未満である．しかし他のすべての実行可能ファイルとの類似度が 0.2 を下回る孤立した実行可能ファイルはない．

19x　バージョン 1.9X

19y　バージョン 1.9Y

表 4　研究 [16] の類似度

	19x	19x o1 md	19x o2 md	19x o2 gl md	19y o2 md
19x db	0.88	0.53	0.48	0.14	0.38
19x		0.55	0.51	0.16	0.39
19x o1 md			0.70	0.22	0.40
19x o2 md				0.29	0.41
19x o2 gl md					0.17

db　実行可能ファイルにデバッグ情報を作成する．
o1　サイズで最適化
o2　速度で最適化
gl　すべてのモジュールにまたがって最適化を行う．
md　ランタイムライブラリを動的にリンクする．

4.2　リンカーオプション

4.1 節の小細工は「19x o2 md」で固定し，リンカーのオプションを変更して異なる実行可能ファイルを作成した．本節の小細工では，実行されるコードは変わらないが，出力されるファイルの構造が異なる．

bs　イメージのベースアドレスを 500000h に変更する．（デフォルトは 400000h である）
fx　再配置テーブルを削除する（常にイメージのベースアドレスにロードされる）．
mg　セクションを結合する．

4.3　ランタイムライブラリ

tmaid は Windows の API だけを使用しているので，ランタイムライブラリをリンクしないこともできる．また，ランタイムライブラリの代わりに自作ライブラリを使用することもできる．4.2 節と同様に 4.1 節の小細工は「19x o2 md」で固定し，この 2 つの実行可能ファイルを作成した．

nl　ランタイムライブラリをリンクしない．
il　自作ライブラリとリンクする．

4.4　ヘッダとファイル構造

本節では 4.2 節で作成した小細工「19x o2 md」の実行可能ファイルを改変した．本節の小細工では，実行されるコードは変わらないが，ヘッダの内容やファイルの構造が異なる．なお，al と rb，ex と sh の 2 組は反対の小細工なので同時に適用できない．

al　ファイルのセクションの境界をメモリにロードされるときの境界に合わせる．
rb　MS-DOS スタブとデバッグ情報を削除する．可能ならばセクションのサイズを減らし，セクションのオフセットを詰める．
ck　PE ヘッダのチェックサムをランダムな値に設定する．

表 5　デジタル署名無効化

アドレス		値	ニーモニック
40E78Fh	改変前	85h C0h	text eax, eax
	改変後	09h C0h	or eax, eax

ex　ファイルの末尾にランダムなデータを追加してファイルサイズを増やす．
sh　ファイルの末尾の未使用領域を削除してファイルサイズを減らす．
pe　PE ヘッダの変更可能な値（タイムスタンプ，リンカーバージョン，イメージバージョン）を 0 にする．
rd　ファイルのヘッダやセクションの間の未使用領域をランダムな値に変更する．

4.5　複合

これまでは各小細工を 1 つづつ適用してきた．本節では 4.3 節と 4.4 節のすべてまたは 1 つを除く小細工を適用することで，複数の小細工が同時に適用された実行可能ファイルを検証する．

4.6　コードサイニング証明書

コードサイニング証明書の効果を検証するために，4.1 節の小細工「19x o2 md」と，4.5 節の小細工「19x o2 md nl rb ck sh rd」に対してコードサイニング証明書でデジタル署名した．本節の実験だけ 2018 年 8 月 6 日に行ったため，デジタル署名前の実行可能ファイルも再検査した．

ct　コードサイニング証明書でデジタル署名する．
br　表 5 に示す同じ動作をする異なる命令でコードの改変を行いデジタル署名を無効化する．

5.　考察

5.1　著者らが行った対策の考察

本節では 3 章の事例から，ソフトウェア作者の対策について考察する．

5.1.1　アンチウイルスベンダーへの報告の考察

3.1 節の PhishWall クライアントの事例では，アンチウイルスベンダーへの報告を主たる対策とした．報告は 3.1.1 項に記述するとおり，手間がかかるものであった．各アンチウイルスベンダーで統一された方法はなく，それぞれのアンチウイルスベンダーが提示する方法に則って報告を行った．対応までに何度も報告を行う必要があり，最初の報告から対応までには時間を要した．

この対策の結果，誤検知はなくなった．

5.1.2　誤検知数の増加

3.2 節の自作ソフトウェアの事例では，特に対策を行わなかった．誤検知数がおおよそ 5 件を超えると，時間の経過とともに誤検知するアンチウイルスソフトウェアが増加する傾向にあった．誤検知数がそれ以下のときには，概ね同

表 6　誤検知結果

節	ラベル	AV01	AV02	AV03	AV05	AV06	AV10	検査日
4.1	19x db							2018 年 7 月 25 日
	19x							
	19x o1 md							
	19x o2 md					✓		
	19x o2 gl md							
	19y db							
	19y							
	19y o1 md							
	19y o2 md							
	19y o2 gl md							
4.2	bs					✓		
	fx							
	mg							
4.3	nl							
	il							
4.4	al					✓		
	rb							
	ck					✓		
	ex					✓		
	sh					✓		
	pe					✓		
	rd							
4.5	19x o2 md al ck ex pe rd							
	19x o2 md al ck sh pe rd							
	19x o2 md rb ck ex pe rd						✓	
	19x o2 md rb ck sh pe rd							
	19x o2 md nl ck ex pe rd							
	19x o2 md nl ck sh pe rd						✓	
	19x o2 md nl al ex pe rd							
	19x o2 md nl al sh pe rd							
	19x o2 md nl al ck pe rd							
	19x o2 md nl al ck ex rd	✓						
	19x o2 md nl al ck ex pe							
	19x o2 md nl al ck ex pe rd							
	19x o2 md nl al ck sh rd	✓						
	19x o2 md nl al ck sh pe							
	19x o2 md nl al ck sh pe rd							
	19x o2 md nl rb ex pe rd			✓	✓		✓	
	19x o2 md nl rb sh pe rd			✓	✓		✓	
	19x o2 md nl rb ck pe rd			✓	✓		✓	
	19x o2 md nl rb ck ex rd	✓		✓	✓		✓	
	19x o2 md nl rb ck ex pe			✓	✓		✓	
	19x o2 md nl rb ck ex pe rd			✓	✓		✓	
	19x o2 md nl rb ck sh rd	✓		✓	✓		✓	
	19x o2 md nl rb ck sh pe			✓	✓			
	19x o2 md nl rb ck sh pe rd			✓	✓		✓	
	19x o2 md il ck ex pe rd						✓	
	19x o2 md il ck sh pe rd						✓	
	19x o2 md il al ex pe rd						✓	
	19x o2 md il al sh pe rd							
	19x o2 md il al ck pe rd							
	19x o2 md il al ck ex rd						✓	
	19x o2 md il al ck ex pe							
	19x o2 md il al ck ex pe rd						✓	
	19x o2 md il al ck sh rd							
	19x o2 md il al ck sh pe							
	19x o2 md il al ck sh pe rd							
	19x o2 md il rb ex pe rd	✓					✓	
	19x o2 md il rb sh pe rd	✓					✓	
	19x o2 md il rb ck pe rd	✓					✓	
	19x o2 md il rb ck ex rd	✓					✓	
	19x o2 md il rb ck ex pe	✓						
	19x o2 md il rb ck ex pe rd	✓					✓	
	19x o2 md il rb ck sh rd	✓						
	19x o2 md il rb ck sh pe	✓						
	19x o2 md il rb ck sh pe rd	✓					✓	
4.6	ct		✓	✓	✓	✓	✓	8 月 6 日
	ct br		✓	✓	✓			
	19x o2 md nl rb ck sh rd	✓		✓	✓		✓	
	19x o2 md nl rb ck sh rd ct	✓			✓			

数で推移した．誤検知数が大きく減少することはなかった．

5.2　事例と実験の考察

本節では 3 章の事例と 4 章の実験の結果を考察する．

5.2.1　小細工の組み合わせ

AV06 以外のアンチウイルスソフトウェアの誤検知は単一の小細工では発生しておらず，小細工が組み合わされることで誤検知が発生すると言える．アンチウイルスソフトウェアによって傾向は異なるが，4.3 節の小細工 nl または il と，4.4 節の rb の組み合わせが誤検知を発生させていることが明らかである．他の小細工は誤検知の原因とは断定できない．

5.2.2　不合理な誤検知

AV06 は同一のソースコードでコンパイラの設定を変更しただけで誤検知する場合がある．AV06 は単一の小細工で誤検知しているが，4.5 節の小細工の複合では誤検知していない．ゆえに，AV06 の誤検知に傾向はなく，おそらく実行可能ファイルのバイナリ列が偶然に何らかのパターンと一致したと思われる．また，AV01 は小細工 nl で小細工 pe を適用した方が誤検知しなくなるという，小細工を重複させた方が誤検知しない結果になった．

5.2.3　誤検知の原因の特定

5.2.1 項の小細工の組み合わせの考察より，自作ソフトウェアの誤検知は，3.2.1 項で推測した自作ライブラリと再構築を同時に適用したことが原因であった．しかし，それらだけが原因ならば，すべての自作ソフトウェアで同じような誤検知数になるはずであるが，図 2 が示すとおり，自作ソフトウェアの中で誤検知数に違いがある．また，自作ライブラリや再構築を行っていない PhishWall クライアントや RPA[3]，「いじくるつくーる」[4] の誤検知は自作ライブラリや再構築では説明できない．

したがって，自作ライブラリや再構築は誤検知の原因の一端にすぎない．5.2.2 項の不合理な誤検知や，小細工以外にも誤検知の原因があると考えられる．

5.3　ファイルの改ざんを検証する仕組みの考察

4.4 節と 4.5 節の結果からチェックサムの正誤はアンチウイルスソフトウェアによる検知とは関係ない．

3.1 節の PhishWall クライアントは元々 EV コードサイニング証明書でデジタル署名されているが誤検知が発生した．

3.2 節の自作ソフトウェアの事例では，各々のアンチウイルスエンジンを 1 つづつ精査すれば，デジタル署名によって誤検知を止めたと言えるアンチウイルスエンジンがあるかもしれない．しかし全体の傾向では，デジタル署名の有無に関係なく誤検知が発生して誤検知数が増加する結果になった．

4.6 節のデジタル署名の実験結果によると，AV03 と AV06 の一部，AV10 のすべてで誤検知がなくなった．デジタル署名によって結果が変わった可能性はある．しかし 5.2.2 項に記述するとおり，AV06 はよくわからない原因で誤検知しており，この変化の原因がデジタル署名なのかは不明である．また，デジタル署名を無効化しても検知または何らかの警告を出すアンチウイルスソフトウェアはなかった．

これらのことから，デジタル署名で誤検知をなくすことはできないと言える．

6. 対策の提案

5.2.1 項で記述する原因となる小細工を行わなければ，誤検知を防げる可能性はある．しかし 5.2.3 項に記述するとおり，5.2.1 項の原因は，誤検知の原因のすべてではない．また，それらの原因となる小細工は，マリシャスである（悪意がある）とは言えない．原因となる小細工を行わないことは，現状のアンチウイルスソフトウェアの実装に対応しただけである．アンチウイルスソフトウェアの仕様が変更されれば，それらは原因ではなくなるかもしれないし，また新たな何かが原因となるかもしれない．しかも 3.1 節の PhishWall クライアントの事例のとおり，アンチウイルスベンダーから情報は得られないので，正しい対策はわからない．結局，ソフトウェアの作成において，ソフトウェア作者がとるべき確実に効果がある対策はない．

しかし，下記に記述することを行う必要があることはわかった．

6.1 素早く報告する

5.1.2 項より，誤検知は放置しておいても解決することはなく，むしろ誤検知数は増加する．ゆえに，誤検知が発生したときには早めにアンチウイルスベンダーに報告しなければ，報告すべきアンチウイルスベンダーの数が増加し，より手間がかかることになる．

6.2 誤検知に備える

5.2.2 項の不合理な誤検知のように，原因が説明できない誤検知もあった．そのことから，どのようなソフトウェアにも誤検知が発生する可能性はあると思われる．特に，使用できなくなったときに甚大な被害が生じるような用途で使われるソフトウェアは，誤検知が発生してもそれを緩和できるような仕組みを備えるべきである．たとえば，

- 可能ならば公開前に Virus Total などで検査を行い，誤検知が発生しないことを確認する．
- 自動的な更新では，新しいファイルが誤検知される可能性を考慮し，古いファイルを保存する．
- 複数の実行可能ファイルで構成されるソフトウェアならば，互いにファイルの存在を確認し，ファイルにアクセスできないときには適切なメッセージを表示する．

などが考えられる．

7. 今後の課題

本研究では小細工と称する実行可能ファイルの改変で，誤検知の原因を調査した．これは主にヘッダの改変などの表層的な改変である．しかしアンチウイルスソフトウェアは，対象の実行可能ファイルのコードを検査しているはずである．そこで，コードのどの部分がマルウェアとして誤検知されているのかを調査することが考えられる．

橋本らは，未検知マルウェアの検体をアンチウイルスベンダーに提供し，対応の早さからアンチウイルスソフトウェアの評価実験を行った [17]．この手法を応用し，誤検知の情報をアンチウイルスベンダーに提供し，対応の早さからアンチウイルスソフトウェアの評価実験を行うことも考えられる．

誤検知の報告は各アンチウイルスベンダー毎に方法が異なるため手間のかかる作業である．一括して報告する方法や報告の自動化などができれば，ソフトウェア作者の負担が軽減すると思われる．

また，6.2 節に記述するように誤検知が発生してもそれを緩和できるような仕組みを検討することが考えられる．

なお，著者の自作ソフトウェアについては，上記の課題を検討しつつ，誤検知の問題を解決する予定である．

参考文献

[1] OSDN 株式会社：プロジェクトファイルリリースの Virus チェック機能に関して，（オンライン），入手先 ⟨https://ja.osdn.net/projects/sourceforge/news/25785⟩（参照 2018-08-03）．

[2] 株式会社セキュアブレイン：PhishWall（フィッシュウォール），（オンライン），入手先 ⟨https://www.securebrain.co.jp/products/phishwall/index.html⟩（参照 2018-08-03）．

[3] 白井　良：十分に発達した RPA はウイルスと見分けが付かない，日経 xTECH/日経 SYSTEMS（オンライン），入手先 ⟨https://tech.nikkeibp.co.jp/atcl/nxt/column/18/00138/071900111/⟩（参照 2018-08-03）．

[4] 矢吹拓也：いじくるつくーるがウイルスバスターによりダウンロードブロックされる件，INASOFT（オンライン），入手先 ⟨https://www.inasoft.org/talk/h201205c.html⟩（参照 2018-08-03）．

[5] 読売新聞：健全サイトを「詐欺」、対策ソフトの誤検知多発，2012 年 12 月 21 日夕刊．

[6] 中村友次郎：INASOFT 矢吹氏、度重なるウイルス誤検知の影響により一部ソフトの更新停止を宣言，Impress Corporation（オンライン），入手先 ⟨https://forest.watch.impress.co.jp/docs/news/545925.html⟩（参照 2018-08-03）．

[7] Virus Total: VirusTotal Monitor, Chronicle Security Ireland Limited (online), available from ⟨https://www.virustotal.com/#/monitor-overview⟩ (accessed 2018-08-03).

[8] Martinez, E.: Launching VirusTotal Monitor, a service to mitigate false positives, Chroni-

cle Security Ireland Limited (online), available from ⟨http://blog.virustotal.com/2018/06/vtmonitor-to-mitigate-false-positives.html⟩ (accessed 2018-08-03).
[9] AV-TEST: Test Modules under Windows, (online), available from ⟨https://www.av-test.org/en/about-the-institute/test-procedures/test-modules-under-windows-usability/⟩ (accessed 2018-08-06).
[10] Virus Bulletin: VB100 Methodology - ver.1.1, (online), available from ⟨https://www.virusbulletin.com/testing/vb100/vb100-methodology/vb100-methodology-ver1-1/⟩ (accessed 2018-08-06).
[11] Kim, D., Kwon, B. J. and Dumitraş, T.: Certified Malware: Measuring Breaches of Trust in the Windows Code-Signing PKI, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, New York, NY, USA, ACM, pp. 1435–1448 (online), DOI: 10.1145/3133956.3133958 (2017).
[12] 株式会社セキュアブレイン：PhishWall クライアント Firefox、Chrome 版のインストーラにおける DLL 読み込みに関する脆弱性と修正完了に関するお知らせ，（オンライン），入手先 ⟨https://www.securebrain.co.jp/about/news/2018/03/180314.html⟩（参照 2018-08-08）.
[13] 独立行政法人情報処理推進機構：PhishWall クライアント Windows 用 Firefox、Chrome 版のインストーラにおける DLL 読み込みに関する脆弱性，（オンライン），入手先 ⟨https://jvndb.jvn.jp/ja/contents/2018/JVNDB-2018-000025.html⟩（参照 2018-08-08）.
[14] 岩本一樹，和崎克己：マルウェアアンパッキングにおけるランタイムライブラリのコード比較によるオリジナルエントリーポイント検出，電子情報通信学会技術研究報告. ICSS, 情報通信システムセキュリティ：IEICE technical report， Vol. 111, No. 82, pp. 57–62（オンライン），入手先 ⟨https://ci.nii.ac.jp/naid/110008746366/⟩ (2011).
[15] 岩本一樹，和崎克己：文書型マルウェアに対するエントロピーとエミュレーションを用いたシェルコード特定方法，情報処理学会論文誌，Vol. 56, No. 3, pp. 892–902 (2015).
[16] 岩本一樹，和崎克己：静的解析により抽出された API 推移に基づくマルウェアの分類，情報処理学会論文誌，Vol. 54, No. 3, pp. 1199–1210 (2013).
[17] 橋本遼太，吉岡克成，松本　勉：未検知マルウェアへの対応に基づくアンチウイルスソフトウェアの評価，研究報告コンピュータセキュリティ（CSEC），Vol. 2012, No. 5, pp. 1–8（オンライン），入手先 ⟨https://ci.nii.ac.jp/naid/170000069611/⟩ (2012).