情報処理学会論文誌 Vol.60 No.9 1477–1488 (Sep. 2019)

ユーザのコンテンツ編集操作を考慮した暗号化型ランサムウェア検知の検討

本多 俊貴[1] 向山 浩平[1] 白井 丈晴[1] 大木 哲史[1] 西垣 正勝[2,a)]

受付日 2018年12月10日, 採録日 2019年6月11日

概要：暗号化型ランサムウェアによる被害が増加している．一般ユーザにとっては，「ランサムウェアが検知を回避しにくい」，「ユーザ環境でのリアルタイムな検知が可能である」という 2 つの特徴を備える検知方式が有用である．そこで本研究では，人間によるコンテンツの編集操作の特徴をホワイトリストとして用いることによって，検知回避困難性とリアルタイム検知性をともに満たす暗号化型ランサムウェア検知方式について検討する．本論文では，その第一歩として，コンテンツ編集時におけるファイル内容表示の有無によって暗号化型ランサムウェアを検知し，被害を未然に防ぐ方式を提案し，有効性評価を行った．

キーワード：暗号化型ランサムウェア，ファイル保護，コンテンツ編集操作，人間らしさ

A Crypto Ransomware Detection Considering User's Content Editing

Toshiki Honda[1] Kohei Mukaiyama[1] Takeharu Shirai[1] Tetsushi Ohki[1]
Masakatsu Nishigaki[2,a)]

Received: December 10, 2018, Accepted: June 11, 2019

Abstract: The number of victims suffering from crypto ransomware is increasing. In order to protect users, a detection method should be able to detect ransomware in the user's real-time environment and make it difficult for the ransomware to avoid detection. This paper proposes a detection method that satisfies these requirements by using human content-editing characteristics as a whitelist. We evaluate the effectiveness of our prototype method, which inspects the consistency between displayed documents and the user's editing operations.

Keywords: crypto ransomware, file protection, content editing, humanness

1. はじめに

2017 年以降，ランサムウェアによる被害が世界中で急増している．McAfee による報告では 2018 年第 2 四半期には 1,800 万件弱のランサムウェアが観測されている [1]．また，ランサムウェアによる全体の被害総額は 2015 年には約 3 億ドルだったのに対し，2019 年には 115 億ドルに増加するとの予測もされている [2]．暗号化型ランサムウェアは，ユーザに隠れて，ユーザの操作とは無関係にファイルの暗号化を行っていく．それに対し，人間は画面上に表示されるコンテンツの内容を確認しながら編集操作を行うことから，人間の通常のコンテンツ編集操作とは異なる挙動が現れる．そこで，ランサムウェアの検知に「人間らしさ」と「ランサムウェアらしさ」による判定の概念を導入するとともに，人間とランサムウェアによるコンテンツ編集操作の特徴の差異に着目することで，検知回避困難性（ランサムウェアが検知を回避しにくい）とリアルタイム検知性（ユーザ環境でのリアルタイムな検知が可能である）を有した暗号化型ランサムウェア検知を提案する．提案方式を実装し，暗号化型ランサムウェアを使用した有効性評価実験を行い議論する．

1 静岡大学大学院総合科学技術研究科
Graduate School of Integrated Science and Technology, Shizuoka University, Hamamatsu, Shizuoka 432–8011, Japan
2 静岡大学創造科学技術大学院
Graduate School of Science and Technology, Shizuoka University, Hamamatsu, Shizuoka 432–8011, Japan
a) nisigaki@inf.shizuoka.ac.jp

© 2019 Information Processing Society of Japan

2. ランサムウェアの分類

近年で感染が報告されているランサムウェアは，大きく分けて「画面ロック型ランサムウェア」と「暗号化型ランサムウェア」の 2 種類に分類される [3]．本章では，それぞれの特徴について説明する．

2.1 画面ロック型ランサムウェア

画面ロック型ランサムウェアは，感染した PC の画面をロックすることで被害者による操作を不能にし，その解除を条件に金銭を要求するランサムウェアである．この場合，感染した PC の OS から独立して動作する駆除ツールを利用することで，ランサムウェアを停止・除去することができる [4]．また，PC をセーフモードで起動させることにより，サードパーティ製ソフトウェアの自動起動を制限することができるため，画面ロック型ランサムウェアの起動を回避し，ランサムウェアを除去することが可能な場合もある [5]．

2.2 暗号化型ランサムウェア

暗号化型ランサムウェアは，ファイルやハードディスクの暗号化を行い，暗号化されたデータの復号を条件に金銭を要求するランサムウェアである．攻撃の概要は以下のとおりである．

1. ドライブバイダウンロード攻撃やランサムウェアのダウンローダをメール添付するなどにより，ランサムウェアを被害者の PC に感染させる．
2. 感染後，ユーザに気づかれないようにバックグラウンドで PC 内のファイルを暗号化する．
3. ファイルの暗号化が完了した後，デスクトップの壁紙変更，脅迫画像ファイル・脅迫テキストファイルなどの生成を通じて脅迫文と金銭の支払い方法をユーザに提示する．

ファイルが暗号化されているため，復号鍵のない状態ではファイルの復元が不可能である．各セキュリティベンダや No More Ransom Project [6] によって暗号化型ランサムウェアが解析され，復号ツールが公開されている場合もあるが，すべての暗号化型ランサムウェアに対応しているわけではない．また，巧妙なランサムウェアはシャドウコピーを格納する VSS ファイルの削除も同時に行うため，PC を感染前の状態に復元することによってファイルを復旧することも不可能である [7]．

このように暗号化型ランサムウェアは，画面ロック型と比較して被害からの回復が非常に困難である．また，近年感染が報告されているランサムウェアによる被害は，Locky や WannaCry，Globeimposter といった「暗号化型ランサムウェア」が多くを占めている [8]．このため本論文では，暗号化型ランサムウェアの対策について検討する．

3. 既存研究とその課題

3.1 暗号化 API の監視による対策

暗号化型ランサムウェアの暗号化処理時に発生する API コールに着目してランサムウェアを検知する方式が重田らによって提案されている [9]．重田らは，Locky や CryptoWall といった代表的なランサムウェアが Microsoft CryptoAPI や OpenSSL の暗号化ライブラリを用いて暗号化を行うことに着目している．この方式では，プログラム実行中の API コールを監視し，前述の暗号化ライブラリの利用を検知することでランサムウェアの検知を行っている．しかし，ランサムウェアが独自に実装した暗号化ルーチンや，無名の暗号化ライブラリを使用することによって，検知の回避が可能である．

3.2 CryptoDrop

Scaife らの CryptoDrop [10] は，暗号化前後のデータに着目し，暗号化型ランサムウェアを早期検知する方式である．具体的には，「マジックナンバによるファイルタイプの確認」，「暗号化前後のファイルの類似度の確認」，「暗号化後のファイルのエントロピーの上昇」の 3 つの主な指標と，「ファイルの削除」，「プロセスが読み書きしているファイルタイプの数」の 2 つの補助的な指標によってランサムウェアの実行を検知する．しかしこの方式は，マルウェアがファイルの一部分のみに暗号化を限定したり，ファイルを分割して入れ替えたりすることにより，類似度やエントロピーを保たせつつファイル構造を破壊することによって，検知を回避できる可能性がある．

3.3 UNVEIL

Kharraz らの UNVEIL [11] は，マルウェアのファイルアクセスおよび画面変化の特徴から暗号化型ランサムウェア検知を行う．まず，解析環境を自動生成し，暗号化の標的となるファイルの生成を行う．次に，解析対象のプログラムを解析環境で動作させ，「書き込みや削除に関する複数の I/O リクエスト」，「同一ファイルの読み込み時と書き込み時のデータバッファ間におけるデータのエントロピーの著しい上昇」，「高いエントロピーを持つ新規ファイルの生成」などの特徴の有無を確認する．また，ランサムウェア感染後には脅迫文が表示され画面表示に変化が大きいことから，感染前後のスクリーンショットを比較する．これらの特徴が顕著に現れるプログラムを暗号化型ランサムウェアとして検出する．しかし，この方式も CryptoDrop と同様，ファイル構造を破壊するという手段をとることでマルウェアは検知を回避できる可能性が残る．また，この方式は解析環境上の検知方式として提案されている．事前に解析環境にて検査を済ませたプログラムでなければ実行を許可しないようにしている場合でも，任意コードが実行でき

© 2019 Information Processing Society of Japan

るような脆弱性が PC に存在する場合 [12] は，その脆弱性を突くことによって検査が完了していないランサムウェアを起動させることができるため，解析環境による検査がバイパスされてしまう可能性も存在する．

4.　提案方式

4.1　コンセプト

本章では提案方式について説明する．提案方式は，2.2 節で述べたとおり，近年被害が顕著であり，感染後にファイルの復元が困難であると考えられる暗号化型ランサムウェアを対象とする．また，3 章の既存研究の分析から，暗号化型ランサムウェア検知方式が検知回避困難性（ランサムウェアが検知を回避しにくい）とリアルタイム検知性（ユーザ環境でのリアルタイムな検知が可能である）を備えることが有用であると考えられる．そこで本研究では，人間のコンテンツ編集操作の特徴をホワイトリストとして用いることによって，これらの特徴を有する暗号化型ランサムウェア検知方式について検討する．本論文では，その第一歩として，コンテンツ編集時におけるファイル内容表示の有無によって暗号化型ランサムウェアを検知し，被害を未然に防ぐ方式を提案する．

検知回避困難性については，ホワイトリスト型の検知方式を採用することによって改善を図る．ランサムウェアの特徴を検知対象とするブラックリスト型の検知方式である場合，ランサムウェアはブラックリストから外れる攻撃手法を 1 つ発見しさえすれば検知を回避できる．一方，ホワイトリスト型の検知方式の場合は，ホワイトリストから外れる攻撃手法はすべて検知されるため，ランサムウェアがとりうる検知回避手段が大幅に制限されることになる．

リアルタイム検知性については，ユーザ利用環境で動作し，ファイルが暗号化される前にランサムウェアを検知する方式が必要である．そこで本論文では，人間の操作の特徴をホワイトリストとして用いる方法を採用する．人間のコンテンツ編集操作をホワイトリストとする検知方式は，ユーザの実際の操作の有無を検査する形態となるため，ユーザ利用環境での稼働を基本とした検知方式となる．

4.2　「人間らしさ」と「ランサムウェアらしさ」

ユーザがファイルを編集する場合，画面に編集用ソフトウェアのウィンドウが表示され，そのウィンドウ上で編集操作を行う．たとえば，テキストファイルであれば画面上にテキストエディタのウィンドウが表示され，表示されたファイルの内容を確認しつつ編集するという操作となる．**図 1** にテキストエディタを用いてコンテンツ（文書）を編集している様子を示す．ユーザによるコンテンツの編集が行われている間，図 1 のようにテキストエディタのウィンドウの内部に編集対象のファイル内容が表示される．本論文では，これを「人間らしさ」を表す特徴として定義する．

図 1　ユーザによるテキストファイル編集時の画面例
Fig. 1　Example of display when user is editing text file.

これに対し，暗号化型ランサムウェアは，ファイルの暗号化を完了する前にユーザに発見されて駆除されてしまうと目的が達成できないため，ユーザに隠れてファイルの暗号化を行う必要がある．つまり，編集対象のファイルの内容が画面上に表示されることがない．本論文では，これを「ランサムウェアらしさ」を表す特徴として定義する．

このように，コンテンツ編集時に画面（ウィンドウ）上にファイルの内容が表示されているか否かという点で，人間とランサムウェアの特徴の差が現れる．提案方式は，この特徴の差を用いて，ランサムウェアのリアルタイム検知を行い，暗号化処理を未然に防ぐことでファイルの保護を実現する．

なお，ファイルに対する操作を行う主体としては，ユーザ（人間）とランサムウェアだけでなく，PC（OS）の中で稼働している正規の自動ソフトウェアも多数存在する．自動ソフトウェアは，PC（OS）の機能を維持するために，あるいは，各種アプリケーションソフトウェアの機能を支援するために，必要とされるファイル操作をそのつど，自律的に実行する．すなわち，自動ソフトウェアによるファイル操作には「人間らしさ」は認められず，提案方式の誤検知対象となる．しかし，自動ソフトウェアが取り扱うファイル群は，ユーザが作成した著作コンテンツ（ユーザが作成したドキュメントファイルや，ユーザが撮影した写真画像ファイルなど，ユーザが PC 画面に表示して閲覧・編集するデータ）とは異なることが通常である．著作コンテンツではないファイル群がランサムウェアによって暗号化されてしまったとしても，OS や各種アプリケーションソフトウェアを再インストールすればそれらのファイル群は復元される．そのため，本論文では，正規の自動ソフトウェアによるファイル操作については議論の対象から外している．

4.3　検知方式

4.3.1　人間による編集操作の特徴

本論文では，人間のコンテンツ編集操作の特徴を以下の 2 点として定義する．4.3.2 項で後述する検知タイミングに

© 2019 Information Processing Society of Japan

図 2　Aero スナップの最小ウィンドウサイズ（右上部）
Fig. 2　Minimum window size of the Aero snap.

おいて，下記のどちらか一方でも確認できない場合はランサムウェアとして検知する．

(a) 画面上に編集用ソフトウェアのウィンドウサイズが，「人間が容易に認知できるサイズ」以上の大きさで表示されている．
(b) 画面上（編集ソフトウェアのウィンドウ内）にファイルの内容が表示されている．

(a) の「人間が容易に認知できるサイズ」は，Windows OS においてウィンドウサイズを自動設定する Aero スナップ機能 [13] を参考にした．Aero スナップ機能では，ウィンドウを画面右上部（左上部，左下部，右下部）に移動させた場合に，ウィンドウサイズが画面の 1/4 に自動設定される（**図 2**）．本論文では，このウィンドウサイズを「容易に認知できるウィンドウサイズ」であると規定し，ユーザが編集作業する際の最小サイズとして設定した．

(b) の「画面上にファイルの内容が表示されている」か否かの検査は，ファイルの種類により検知方式が異なる．

WYSIWYG 型ファイル：

WYSIWYG（What You See Is What You Get）型のファイルでは，表示する内容およびフォント情報やレイアウト情報などがファイルに含まれており，編集ソフトウェアがそれらの情報を用いてファイルの内容を表示する．画像ファイルや Microsoft Word ファイルがその典型例である．WYSIWYG 型ファイルは，ファイル内の情報からコンテンツ全体を再構築することができるので，「ファイルから再構築したコンテンツ全体の画像」の中に「コンテンツ編集時の（編集ソフトウェアのウィンドウ内の）画面」が含まれるか否かを画像マッチングによって確認することによって，(b) の検査が可能である．

非 WYSIWYG 型ファイル：

WYSIWYG 型でないファイルでは，フォント情報やレイアウト情報などがファイルに含まれておらず，編集ソフトウェアのウィンドウサイズや設定（例：「右端で折り返す」や「表示に利用するフォント」など）に依存して画面の表示が変化する．テキストファイルがその典型例である．「ファイル内に記録されている文字列」の中に「コンテンツ編集時の実際の（編集ソフトウェアのウィンドウ内の）画面から抽出された文字列」が含まれているか否かを文字列マッチングによって確認することによって，(b) の検査が可能である．

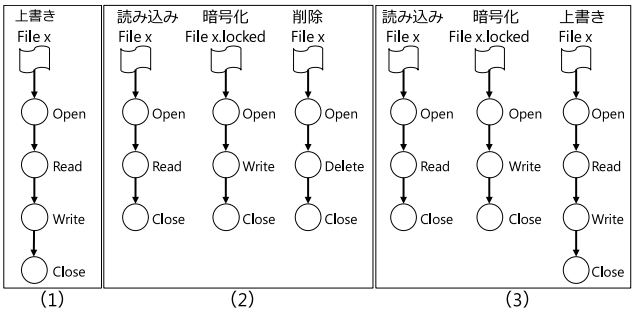


図 3　暗号化型ランサムウェアのファイルへのアクセスパターン
Fig. 3　File access patterns by crypto ransomware.

4.3.2　検知タイミング

暗号化型ランサムウェアによるファイルの暗号化時に発生するファイルへのアクセスパターンは，**図 3** の 3 つのタイプに大別される [11]．以下，それぞれのタイプについて説明する．

(1) 攻撃対象となるファイル x を読み込み，ファイルの内容を暗号化したうえで，その内容をファイル x に上書きする．
(2) 攻撃対象となるファイル x を読み込み，ファイルの内容を暗号化したうえで，その内容を新たなファイル x.locked として生成する．その後，ファイル x を削除する．
(3) 攻撃対象となるファイル x を読み込み，ファイルの内容を暗号化したうえで，その内容を新たなファイル x.locked として生成する．その後，ファイル x を別データで上書きして破壊する．

これらすべてのタイプにおいて，「ファイル x の削除あるいは上書き」によって攻撃対象となるファイル x の破壊が行われている．したがって，「ファイルの削除あるいは上書き」が発生した時点で 4.3.1 項の (a) および (b) がすでに満たされていることを検査し，確認できなかった場合はその「ファイルの削除あるいは上書き」を禁止することで，ファイルの暗号化が未然に防止される．以上より，4.3.1 項で説明した特徴をホワイトリストとして用い，暗号化型ランサムウェアによる「ファイルの削除あるいは上書き」を監視ならびに禁止することで，ランサムウェアのリアルタイム検知が実現される．

なお，暗号化の前後にファイルのリネームを行うランサムウェアの存在が確認されている．そのため，上記の「ファイル x の削除あるいは上書き」については，「ファイル x あるいは “ファイル x のリネームによって生成されたファイル x.renamed” の削除あるいは上書き」の発生という形で拡大解釈して検知を行う必要がある．

© 2019 Information Processing Society of Japan

5. 提案方式の実装

提案方式の第一歩として，本論文では WYSIWYG 型のファイルである画像ファイル（BMP，JPEG，PNG ファイル）と，非 WYSIWYG 型のファイルであるテキストファイル（TXT ファイル）を対象とした暗号化型ランサムウェア検知プログラムを実装した．実際にはランサムウェアによって生成されるすべてのプロセスを対象とした監視および検知を行う必要があるが，今回の実装は簡単化のため，単一プロセスで動作するランサムウェアの検知を対象としている．

5.1 画像ファイルの画面表示判定

画像ファイルの内容が画面上に表示されているかどうかを判定するために，以下の 2 つの処理が必要である．

1. 画面表示の取得
2. 1 で取得した画面表示と画像ファイルの比較
3. 2 の結果からの類似度の算出

以下，それぞれの動作の実装について述べる．

5.1.1 画面表示の取得

画像編集ソフトによって画像ファイル A が開かれた際の表示画面をスクリーンショットでビットマップ画像 B として取得する．たとえば，Microsoft Paint（mspaint.exe）で画像ファイル「Lenna.jpg」を開いた場合のスクリーンショット画像は図 4 右のように取得される．

5.1.2 画面表示と画像ファイルの比較

画像ファイルが画面上に表示されているかどうかを判定するために，画像マッチングを行う．画像ファイル A と 5.1.1 項で取得した画面表示のスクリーンショット画像 B との間で画像マッチングを行う（図 4）．本論文では，この画像マッチングに OpenCV [14] を利用し，特徴点ベースの画像マッチングを用いる．具体的には，(i) AKAZE アルゴリズムによって画像 A と B の特徴点をそれぞれ抽出し，(ii) K 最近傍法（$k = 2$）によりマッチする特徴点の候補を抽出した後に，(iii) 係数を 0.5 とした ratio test によりマッチ判定を行い，(iv) マッチした特徴点を見つける．

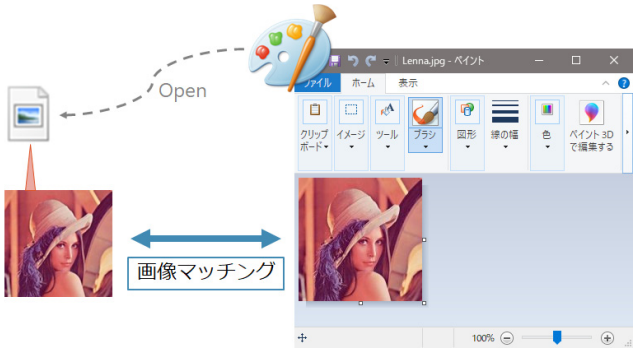


図 4　画像ファイルの内容の画面表示判定の例

Fig. 4　Example of how to check whether image content is displayed.

なお，Microsoft Office や Adobe Acrobat などの文書ファイル（DOCX ファイル，PPTX ファイル，PDF ファイルなど）の場合は，文書ファイルを変換することによってビットマップ画像化することができる．これを画像ファイル A として取り扱えば，これら文書ファイルについても，同様の方法で比較が可能であると考えている．本論文においては，この機能自体は未実装であるが，Universal Office Converter [15] や ImageMagick [16] を利用することによって文書ファイルのビットマップ画像化が可能であることについては，実際に確認済みである．

5.1.3 画面表示と画像ファイルの類似度の算出

5.1.2 項のアルゴリズムによってマッチした特徴点の数を画像 A と画像 B の類似度として用い，その数を閾値と比較することによって，画像ファイルが画面表示されているか否かの判定を行う．今回は，この閾値は以下の予備実験により決定した．まず，20 個の画像ファイルを Microsoft Paint（mspaint.exe）で読み込み，5.1.2 項のアルゴリズムを用いてマッチした特徴点の数を計数した．その結果，マッチ数の最低値は 10 であった．最低値を記録したファイルは 150 px × 150 px の画像ファイルであり，他のファイルに比べて解像度が低く，抽出できた特徴点が少ないという状況であった．そこで今回はその画像ファイルを最低基準とし，画像 A と画像 B がマッチしているか否かを判定する閾値を 10 に設定した．

なお，画像のサイズが大きい場合や解像度が高い場合など，画面上（画像編集ソフトのウィンドウ内）に画像ファイルの全体が表示されない場合もある．また，複数ページからなる文書ファイルの場合も，画面上（文書編集ソフトのウィンドウ内）に表示されるのは全体の文書ファイルの一部となる．特徴点ベースのマッチングであれば，大きなサイズの画像の中に含まれる小さなサイズの部分画像を見つけることも得意であり，今回はマッチした特徴点数で判別していることから，画像や文書の一部分のみが画面に表示されている場合でも，特徴点が十分に存在すれば正しく判定することが可能である．

5.2 テキストファイルの画面表示判定

テキストファイルの内容が画面上に表示されているかどうかを判定するために，以下の 3 つの処理が必要である．

1. 画面上に表示された文字列の取得
2. 1 で取得した文字列とファイル内容の比較
3. 2 の結果からの類似度の算出

以下，それぞれの動作の実装について述べる．

5.2.1 画面上に表示された文字列の取得

画面上に表示された文字列を取得するためには，画面情報をスクリーンショットで画像として取得し，その中に映っている文字列を文字データに変換する必要がある．本

© 2019 Information Processing Society of Japan

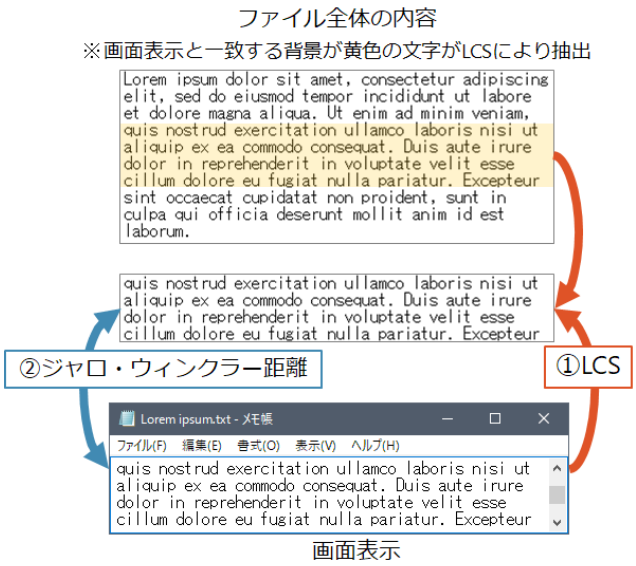


図 5　テキストファイルの内容の画面表示判定の例

Fig. 5　Example of how to check whether file content is displayed.

論文では，これに Tesseract OCR [17] を用いる．ただし，OCR の特性として，OCR で取得する文字列は確実に正確であるとはいえず，表示されている文字とは異なる文字の取得や，ファイルの内容以外の文字列（たとえば，ソフトウェアのツールバーに記されている文字）も抽出されてしまうという欠点が存在する．

5.2.2　取得した文字列とファイル内容の比較

長文のファイルである場合，文書ファイルの内容のすべてが画面上に一度に表示されることはなく，ファイルの内容の一部分のみが画面上に表示されることになる．このため，画面上から取得した文字列とファイル内容の比較を行う前に，「現在，画面上に表示されている内容がファイルの内容のどの部分に一致しているのか」を判別する必要がある．本論文では，この判別に Diff の LCS（Longest Common Subsequence）[18] を用いる．**図 5** の ① に，ファイル全体の内容の中から画面表示されている部分と一致している箇所（背景が黄色の文字列）が，LCS によって発見された様子を示す．

5.2.3　取得した文字列とファイル内容の類似度の算出

OCR を用いて画面上から取得した文字列（5.2.1 項）と LCS を用いて発見したファイル内の文字列（5.2.2 項）の類似度の計算には，文字列どうしの近さを 0〜1 の間で数値化するジャロ・ウィンクラー距離を用いる [19]．ジャロ・ウィンクラー距離は，数値が大きいほど距離が近いことを表す．図 5 の ② に，OCR によって取得した文字列と LCS によって抽出した文字列の間のジャロ・ウィンクラー距離を算出している様子を示す．

算出されたジャロ・ウィンクラー距離に対して閾値判定を行うことで，ファイルの内容が画面上に表示されているか否かを判別する．今回は，この閾値は以下の予備実験により決定した．まず，20 個のテキストファイルをメモ帳（notepad.exe）で読み込み，画面上の表示と当該ファイルの内容とのジャロ・ウィンクラー距離を計算した．その結果，距離の最低値は約 0.78 であった．次に，読み込まれたファイルとは異なる内容を画面表示している場合のジャロ・ウィンクラー距離を計算した．その結果，距離の最高値は約 0.66 であった．以上より，ファイルの内容が画面上に表示されているか否かを判定する閾値を 0.7 に設定した．

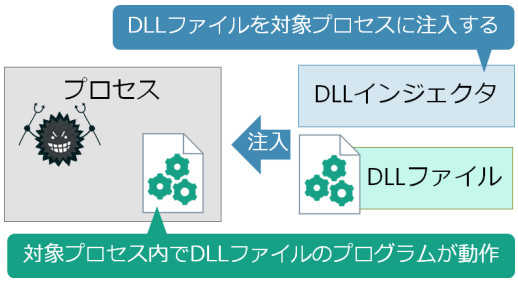


図 6　検知プログラムの構成

Fig. 6　Structure of detection program.

5.3　検知プログラム

検知プログラムはファイル操作を行っているソフトウェアの監視・制御を行う．5.1 節および 5.2 節に示した処理を用いて暗号化型ランサムウェアであるかどうかの判定をし，暗号化型ランサムウェアである場合には，当該ソフトウェアによるファイルの削除あるいは上書きを禁止する．

ファイルへのアクセスの監視は，DLL インジェクション [20] により，監視対象のソフトウェアの API コールを監視することによって実現した．DLL ファイルを埋め込む役割を担う「DLL インジェクタ」と埋め込み先のプロセス内で動作するランサムウェア検知処理が記述された「DLL ファイル」の 2 つから構成される（**図 6**）．

5.3.1　DLL インジェクタ

DLL インジェクタはプロセスに DLL ファイルを埋め込む動作をする．今回の DLL インジェクタは，新規プロセスの生成を 0.5 秒ごとに監視し，新規プロセスに対して DLL ファイルを注入する．

5.3.2　DLL ファイル

DLL ファイルに記述されている検知プログラムの処理内容の概要を次に示す．

① DLL 埋め込み先のプロセスによるファイル読み込みが発生した場合，検知プログラムは当該ファイルにアクセスし，当該ファイルの形式を検査する．

② 当該ファイルが WYSIWYG 型であった場合，③ に進む．当該ファイルが非 WYSIWYG 型であった場合，⑦ に進む．

③ 検知プログラムも当該ファイルにアクセスし，当該ファイルをビットマップ画像に変換する．

④ ファイル読み込みを実行したプロセスの表示ウィンド

© 2019 Information Processing Society of Japan

ウの画面を，スクリーンショットによってビットマップ画像として取得する．その際，表示ウィンドウのサイズが，4.3.1 項で定めた「人間が容易に認知できる大きさ」を超えているか確認する．

⑤ ③で取得した画像と④で取得した画像を，5.1 節で示した方法を用いて画像マッチングすることにより，プロセスが読み込んだ画像ファイルが画面上に表示されているかを判定する．

⑥ ④のウィンドウサイズが，人間が容易に認知できる大きさを超えており，かつ，⑤でファイルの内容が画面上に表示されていることが確認できた場合に限り，人間によるコンテンツ編集操作であると判定する．⑪に進む．

⑦ 検知プログラムも当該ファイルにアクセスし，当該ファイルに含まれる文字列を取得する．

⑧ ファイル読み込みを実行したプロセスの表示ウィンドウの画面を，スクリーンショットによって画像として取得し，OCR を用いて画像中に表示されている文字列を抽出する．その際，表示ウィンドウのサイズが，4.3.1 項で定めた「人間が容易に認知できる大きさ」を超えているか確認する．

⑨ ⑦で取得した文字列と⑧で抽出した文字列を，5.2 節で示した方法を用いて比較し，プロセスが読み込んだテキストファイルが画面上に表示されているかを判定する．

⑩ ⑧のウィンドウサイズが，人間が容易に認知できる大きさを超えており，かつ，⑨でファイルの内容が画面上に表示されていることが確認できた場合に限り，人間によるコンテンツ編集操作であると判定する．

⑪ 人間によるコンテンツ編集操作であることが確認できた場合は，「①のファイル読み込みを実行したソフトウェア」による「①で読み込んだファイル」への上書きや削除を許可する．確認できなかった場合は，ファイルへの上書きと削除を拒否する．

具体的には，次に示す API をフックし，監視・ファイル制御の動作を追加することによって，今回の DLL ファイルを実装した．

・ReadFile，CreateFileW（読み込み）：

これらの API がコールされた際に，読み込まれたファイルのパスの拡張子から，WYSIWYG 型ファイル，非 WYSIWYG 型ファイルであることの判別をし（上記①），画像マッチング（上記③～⑥）あるいは OCR マッチング（上記⑦～⑩）を行うことで，読み込まれたファイルが画面上に表示されているか（人間によるコンテンツ編集操作であるか）否かの判断を行う．表示されていることが確認できた場合は，当該ファイルの書き込み/削除の許可フラグを「可」にセットする．表示されていることが確認できない場合は，許可フラグを「非」にセットする．検査対象外のファイルである場合は，許可フラグは「可」とする．

・WriteFile，DeleteFileW（書き込み/削除）：

これらの API がコールされた際に，該当ファイルの書き込み/削除に関する許可フラグが「可」である場合には，書き込み/削除の操作を行う．許可フラグが「非」である場合は，その操作を行わないことでファイルを保護する（上記⑪）．

・MoveFileW，MoveFileExW（リネーム）：

ファイルを読み込む前に当該ファイルの拡張子をリネームするランサムウェアも存在する．ファイルの拡張子が変更されてしまった場合，上記①におけるファイルタイプの確認が正しく機能しない．そこで，これらの API がコールされた際に，リネームされたファイルのリネーム前後のパス情報を記録しておき，ファイル読み込みの際のファイルタイプの検査（上記①）の際に，これを利用する．

5.4 画面表示判定のタイミング

4.3.2 項で述べたように，「ファイルの削除あるいは上書き」が発生した時点でファイル内容の画面表示の有無を検査することが，提案方式の本来の形態である．しかし，画面表示判定のために必要となる OCR 処理や画像マッチングは，処理時間を要するタスクである．著者らが行った予備実験においては，OCR の場合も画像マッチングの場合も最長 3 秒弱の処理時間を要した．そのため，今回の実装においては，5.3.2 項で示したとおり，ファイルを読み込んだ時点で画面表示判定の処理を開始するという方法によって，ユーザビリティの確保に配慮することとした．人間によるコンテンツ編集は，画面を確認しながらの操作となるため，相応の時間が要されることが予想される．この時間を有効活用し，画面表示の検査を実施する．

将来，PC のパフォーマンスが十分に向上した場合には，書き込みが発生した時点で画面表示の判定を行い，「ユーザ（人間）が今まさにファイルを編集している」ことをより明確に確認するようにすべきである．これについては，今後の課題とする．

6. 実験

5 章で実装した検知プログラムを用いて検知実験と誤検知実験を行い，提案方式の有効性を評価した．その結果を以下に示す．

6.1 検知実験

6.1.1 実験目的

検知プログラムが動作した状態でランサムウェアを実行させた際に，ファイルの暗号化が阻止されるか否かを確認することによって，ランサムウェア検知に対する提案方式の有効性を確認する．なお，ファイル名がリネームされてしまった場合でも，ファイル内容が変更されていない場合

© 2019 Information Processing Society of Japan

表 1 実験環境
Table 1 Experiment environment.

仮想マシン	QEMU	1:2.5+dfsg-5ubuntu10.32
	ゲスト OS	Windows 8.1 x86
	画面解像度	1,024×768
	メインメモリ	2,048MB
	インターネット接続	なし
	OS の言語設定	英語
	Windows Defender	無効化

表 2 各種ファイルの格納フォルダと内容物
Table 2 File contents used for experiments.

ランサムウェア検体	C:¥Users¥Owner¥Downloads
検知プログラム	C:¥Program Files¥Prog
保護対象ファイル	C:¥Users¥Owner¥Desktop 画像 20 個，テキスト 20 個
	C:¥Users¥Owner¥Pictures 画像 10 個
	C:¥Users¥Owner¥Documents テキスト 10 個

は「阻止成功」とする．

6.1.2 実験方法

検知実験は仮想 PC ソフトウェアである QEMU 上で行った．実験環境を**表 1** に示す．PC 内に検知プログラムと保護対象ファイルを配置し，検知プログラムを動作させた状態で PC に暗号化型ランサムウェアを感染させる．その際に，ランサムウェアによって攻撃を受けたファイルの数（標的数），および，検知プログラムによって保護することができたファイルの数（保護数）を確認する．

暗号化型ランサムウェアの検体は，Check Point Threat-Cloud Cyber Threat Map [21] において 2018 年上半期の観測数が多かったランサムウェアファミリ [8] に，Cerber, Jaff のランサムウェアファミリを加えたものを収集対象として，Hybrid Analysis [22] で収集した．収集された検体のうち，今回の実験環境で動作が確認できた 12 検体を実験に使用した．また，検体，検知プログラム，保護対象ファイルの配置を**表 2** に示す．なお，画像ファイルは Pixabay [23] のフリー画像および SIDBA [24] の画像を使用し，テキストファイルは BLIND TEXT GENERATOR [25] で生成した文字列をテキストファイルとして保存したものを使用した．

マルウェアの中には，仮想マシンを用いた解析環境を検知して動作を変えるものが存在する．しかし，今回は，複数の検体について実験を行うことから，環境構築（ロールバック）の容易性の観点から，仮想マシンでの実験を選択した．QEMU を選択した理由は，著者らの調べた限りでは，QEMU が他の仮想化ソフトウェアと比べて，比較的，マルウェアの Anti-VM の対象になっていない点などによる．なお，検体実行の際に外部リソースの安全性を担保できないことから，インターネット接続のない隔離環境での実験としている．

表 3 検知実験 実験結果
Table 3 Result of detection experiment.

検体	ファミリ	画像ファイル		テキストファイル	
		標的数	保護数	標的数	保護数
A	Cerber	20	20	2	2
B	Cerber	20	20	2	2
C	GlobeImposter	30	20	0	0
D	Jaff	0	0	0	0
E	Locky	30	0	30	0
F	Locky	30	0	30	0
G	Locky	25	0	30	0
H	WannaCry	30	0	30	0
I	WannaCry	30	30	30	30
J	Cerber	20	20	6	6
K	GlobeImposter	30	29	30	30
L	GlobeImposter	30	20	30	30

6.1.3 実験結果

実験結果を**表 3** に示す．検体 A，B，I，J では，標的となったファイルはすべて保護することができた．しかし，検体 C，K，L では一部のファイルが，検体 E，F，G，H ではすべてのファイルが保護できていない結果となった．また，検体 D を除くすべての検体において，感染によってファイル名のリネームが発生した．そのうち，検体 C，H，K，L については，リネーム後のファイル名に元のファイル名が含まれていた．ファイルの保護ができなかった原因については 7.1 節で，ファイル名のリネームによる影響については 7.2 節で，それぞれ詳述する．

6.2 誤検知実験

6.2.1 実験目的

検知プログラムを動作させた PC 上で，正規ユーザが任意のファイルの編集をした際に，ランサムウェアとして誤検知されないことを確認する．

6.2.2 実験方法

実験環境および各ファイルの配置は 6.1.2 項と同一である．対象となるファイルの種類によって検知の方法が異なっているため，以下の 2 種類の実験を行う．

WYSIWYG 型ファイル：

検知実験で用意した画像ファイルのうち，Pictures フォルダ内の 10 個のファイルを使用する．検知プログラムを起動した状態で，それぞれのファイルを Microsoft Paint（mspaint.exe）で開く．画像に対してブラシツールで微細

© 2019 Information Processing Society of Japan

表 4　誤検知実験　実験結果（WYSIWYG）
Table 4　Result of false detection experiment for WYSIWYG file.

画像ファイル	解像度	マッチした特徴点の数	上書き
A	640×360	422	○
B	960×640	159	○
C	960×691	1108	○
D	960×640	1242	○
E	787×720	400	○
F	518×720	541	○
G	960×640	404	○
H	150×150	34	○
I	150×150	11	○
J	531×340	797	○

表 5　誤検知実験　実験結果（非 WYSIWYG）
Table 5　Result of false detection experiment for Non-WYSIWYG file.

テキストファイル	文字数	JW 距離	上書き
A	1045	0.788077	○
B	2057	0.880777	○
C	1725	0.800500	○
D	1679	0.790589	○
E	1605	0.797696	○
F	1849	0.806512	○
G	1859	0.800596	○
H	2055	0.877522	○
I	1696	0.800023	○
J	1595	0.799662	○

な編集を行い，上書き保存を行う．この際，上書きが許可されることを確認する．

非 WYSIWYG 型ファイル：

検知実験で用意したテキストファイルのうち，Documents フォルダ内の 10 個のファイルを使用する．検知プログラムを起動した状態で，それぞれのファイルをメモ帳（notepad.exe）で開く．文字列の先頭にアルファベット「A」を追加し，上書き保存を行う．この際，上書きが許可されることを確認する．

6.2.3　実験結果

WYSIWYG 型ファイル：

実験結果を**表 4** に示す．「上書き」の列は，上書きに成功した場合が「○」，上書きに失敗した場合が「×」である．この結果より，すべての画像ファイルにおいて，マッチした特徴点は閾値である 10 を超え，人間によるファイル操作であるとして上書きが許可されていることが確認できた．

非 WYSIWYG 型ファイル：

実験結果を**表 5** に示す．「JW 距離」の列はジャロ・ウィンクラー距離を示し，「上書き」の列は上書きに成功した場合が「○」，上書きに失敗した場合が「×」である．この結果より，すべてのテキストファイルにおいて，ジャロ・ウィンクラー距離の閾値である 0.7 を超え，人間によるファイル操作であるとして上書きが許可されていることが確認できた．

7.　課題と展望

7.1　検知漏れの原因

本論文では，検知プログラムの実装に DLL インジェクションを採用した．また，今回は簡単化のため，単一プロセスで動作するランサムウェアの検知を対象とした検知プログラムを実装している．しかし，マルウェアの中には，DLL インジェクションを検知し動作を停止するものや，複数のプロセスを起動して目的を遂行するものも存在する．そのようなランサムウェアには，今回実装を行った検知プログラムは有効に動作しない．実際，6.1 節の検知実験において検知に失敗した検体 D は，検知プログラムを動作させている場合のみ暗号化の動作を行わないことが確認された．したがって，検体 D は DLL インジェクションの発生を検知し，動作を停止させている可能性がある．また，検体 H は，Sleep.exe を何度も呼び出して，複数のプロセスを起動させていた．検体 C，F，G，K，L においても複数プロセスの呼び出しが観測された．

さらに，6.1 節の検知実験の際の動作ログを確認したところ，今回の検知プログラムで監視していた API のみでは，ファイルの Read および Write の監視が不十分であったことが判明した．たとえば，今回の実装では CreateFileW 関数は Read 動作を行う API として扱っているが，引数のパラメータによっては書き込みにも使用できるファイルハンドルを取得できる．実際，6.1 節の検知実験において検知に失敗した検体 E の動作ログの中に，検体 E が CreateFileW 関数を用いて取得したファイルハンドルを利用してファイルの書き換えを行っていると推測される形跡があった．今回の実装では，ファイルハンドルと API の種類のみによる監視を行っているが，API コールの引数を考慮して Read，Write，Delete の判別をする実装を行う必要があることが判明した．また，検体 C，F，G においても API コールの監視不足を裏付けるログが残っていた（すなわち，検体 C，F，G，K，L においては，複数プロセスの呼び出しと API コールの監視不足が複合的に発生していた）．

7.2　検知プログラムの実装

提案方式はホワイトリスト型の検知（ファイルの内容を画面上に表示していなければ，ファイルの上書きや削除が許可されることはない）であるため，そのコンセプトが完

© 2019 Information Processing Society of Japan

全に実装されているのであれば，すべてのランサムウェアに対して有効に作用することが期待される．しかし，今回の検知プログラムの実装においては，7.1 節で述べたような「実装の漏れ（提案方式のコンセプトが完全に実装できなかった部分）」を含んでしまっていた．以上より，提案方式は有効であるものの，今回の検知実験（6.1 節）では実装の不備により検体 C，D，E，F，G，H，K，L に対するファイル保護が達成できなかったと考える．

また，今回の検知実験において，複数の検体がファイルの暗号化とともにファイル名のリネームを行うことが確認された．そして，そのうちの多くの検体において，リネーム後のファイル名から元のファイル名を推測することができなかった．拡張子を含めたファイル名がランサムウェアによって勝手に変更された場合，たとえファイルの内容が保持されていた場合でも，「そのファイルが保有している情報が何なのかが分からない」，「どのソフトウェアで利用できるファイルであるのか判断できない」という状態に陥り，復旧にかかる労力が多大となると考えられる．そのため，今回の検知プログラムの実装に加え，ファイル名の変更（あるいはファイルの移動）を監視・記録し，その情報をもとにしてファイル名についても復元できる機能を備えることが有用であろう．

以上の問題に適切に対処するためには，DLL インジェクションによる API 監視をはじめとした OS の高レイヤにおける検知機能の実装と，ファイル I/O アクセスの検査をはじめとした OS の低レイヤにおける検知機能の実装とを組み合わせる必要がある．そのような OS の両レイヤにまたがった検知方式を OS の機能として実装し，すべてのプロセスを監視することによって，提案方式を完全な形で動作させることが必要であると考える．

7.3　ランサムウェアによる検知の回避

提案方式の検知プログラムは，「画面上にファイル内容（コンテンツ）が表示されていたか否か」によって検知を行っている．したがって，ランサムウェアが提案方式を回避するには，ファイルを暗号化しようとする時点で，対象ファイルの内容を画面上に表示させることが必須となる．これは，ランサムウェアにとっては，自分の存在をユーザに気づかれてしまうリスクを招くことになる．ランサムウェアはこの状況に対し，ユーザに気づかれにくい形で画面上にファイルの内容を表示することで検知を回避するという手段に出ることが考えられる．これに対処するために，提案方式では 4.3.1 項で最小ウィンドウサイズを設定している．そのため，ランサムウェアがユーザに気づかれずに暗号化を行うことは困難になると期待される．さらに，提案方式に「ファイルの内容が画面上により長く表示されている場合ほど人間らしさが大きくなる」といった指標を追加することで，ランサムウェアがユーザに発見されてしまう

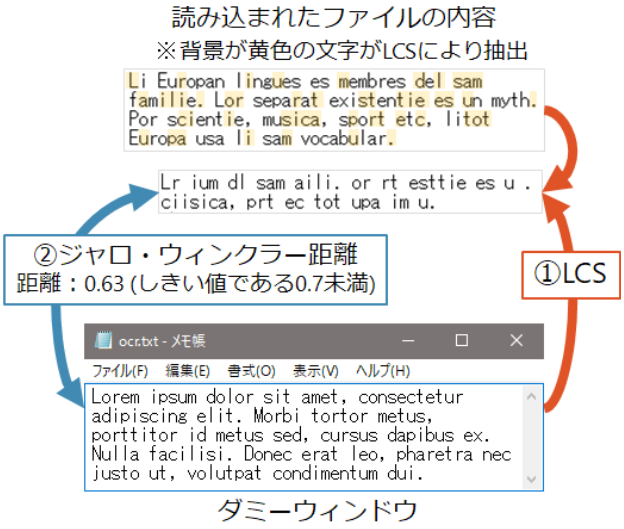


図 7　ダミーファイル表示時のマッチング（非 WYSIWYG）
Fig. 7　Example of distance calculation when dummy file is displayed (Non-WYSIWYG).

確率をより高めることができることも期待される．

また，ランサムウェアがダミーファイルを用意し，画面上にはダミーファイルを表示して，バックグラウンドで複数の攻撃対象ファイルを暗号化するといった手段に出ることも考えられる．しかし，別のコンテンツが画面に表示されている場合には，ファイルの内容と画面に表示されている内容の類似性が低くなる（非 WYSIWYG 型ファイルの例を図 7 に示す）．これにより，ダミーの内容を表示するという検知回避手段は有効には働かないことが期待される．

7.4　ファイル内容を表示しないソフトウェアへの対応

提案方式は，ユーザによるコンテンツ編集操作を，「人間らしさ」を表す特徴として用いるホワイトリスト型のランサムウェア検知方式である．このため，ファイル形式変換ソフトウェアのような「ユーザからの指示を受けてコンテンツの編集を行うが，ファイルの内容を表示せずに編集操作を行うソフトウェア」に対しては誤検知が発生してしまう．しかし，これらのソフトウェアにおいては，ファイルの内容自体は表示されないものの，操作対象となるファイルのファイルパスやファイル名といった情報が画面上に表示されることが通常であると考えられる．また，ユーザからのソフトウェアへの入力（例：操作対象ファイルの指定，実行ボタンのクリック）も発生すると考えられる．これらを「人間らしさ」の定義に含めることで，ファイル内容を表示せずにファイル操作を行うソフトウェアについても保護することが可能になると考えている．

8.　おわりに

本論文では，「人間らしさ」と「ランサムウェアらしさ」の差に着目し，検知回避困難性とリアルタイム検知性を備えた暗号化型ランサムウェア検知方式を提案した．人間に

© 2019 Information Processing Society of Japan

よるコンテンツ編集操作を「人間らしさ」と定義してホワイトリスト型の検知を行うことによって，検知回避困難性を向上させた．ランサムウェアの検知時には，「ファイルの削除あるいは上書き」を制御することによってランサムウェアによるファイルの暗号化を未然に防止する．WYSIWYG 型の画像ファイル（BMP，JPEG，PNG ファイル）と，非 WYSIWYG 型の文書ファイル（TXT ファイル）の保護を対象とした暗号化型ランサムウェア検知プログラムを実装し，有効性評価を行った．検体を用いた検知実験では，検知プログラムの想定どおりに動作するランサムウェアに対してファイルの暗号化を未然に防止できることを確認した．また，正規ユーザの編集操作による Microsoft Paint とメモ帳を用いた誤検知実験においては，検知プログラムによる誤検知が発生しないことを確認した．

今後の課題として，実装に関する問題の改善，ファイル内容が画面に表示されないソフトウェアへの対応に取り組む．

謝辞 WYSIWYG 型ファイルの画像マッチングに用いるアルゴリズムの選定において，京都大学情報学研究科中澤篤志准教授にご助言をいただいた．また，検知プログラムの実装に関して，NTT セキュアプラットフォーム研究所青木一史様，鐘本楊様にご助言をいただいた．謹んで感謝の意を表する．

参考文献

[1] McAfee：McAfee Labs 脅威レポート 2018 年 9 月，入手先 ⟨https://www.mcafee.com/enterprise/ja-jp/assets/reports/rp-quarterly-threats-sep-2018.pdf⟩（参照 2018-11-19）．

[2] Steve Morgan: Global Ransomware Damage Costs Predicted To Hit $11.5 Billion By 2019, Cybercrime Magazine, available from ⟨https://cybersecurityventures.com/ransomware-damage-report-2017-part-2/⟩ (accessed 2018-11-19).

[3] Marvin the Robot：インフォグラフィック：ランサムウェアについて知っておくべきこと，カスペルスキー公式ブログ，入手先 ⟨https://blog.kaspersky.co.jp/ransomware-infographics/13223/⟩（参照 2017-03-27）．

[4] Anastasiya Angel：画面ロック型ランサムウェアへの対処，カスペルスキー公式ブログ，入手先 ⟨https://blog.kaspersky.co.jp/kaspersky-windowsunlocker-2/11601/⟩（参照 2017-03-28）．

[5] Sophos：「警察からの警告」を装う Android ランサムウェア．その回避策と感染した場合の対策について，Sophos News and Press Releases，入手先 ⟨https://www.sophos.com/ja-jp/press-office/press-releases/2014/05/ns-android-police-warning-ransomware.aspx⟩（参照 2017-03-28）．

[6] The No More Ransom Project, available from ⟨https://www.nomoreransom.org/⟩ (accessed 2018-11-19).

[7] Paul Ducklin: Got ransomware? What are your options?, Naked Security, available from ⟨https://nakedsecurity.sophos.com/2016/03/03/got-ransomware-what-are-your-options/⟩ (accessed 2018-11-19).

[8] Check Point: Cyber Attack Trends 2018 Mid-Year Report, available from ⟨https://pages.checkpoint.com/cyber-attack-2018-mid-year-report.html⟩ (accessed 2018-11-19).

[9] 重田貴成，森井昌克，長谷川智久，池上雅人，石川堤一：ランサムウェアにおける暗号化処理について，コンピュータセキュリティシンポジウム 2016 論文集，Vol.2016, No.2, pp.134–137 (2016).

[10] Scaife, N., Carter, H., Traynor, P. and Butler, K.R.B.: Cryptolock (and drop it): Stopping ransomware attacks on user data, *2016 IEEE 36th International Conference on Distributed Computing Systems* (*ICDCS*), pp.303–312, IEEE (2016).

[11] Kharraz, A., Arshad, S., Mulliner, C., Robertson, W. and Kirda, E.: UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware, *25th USENIX Security Symposium*, pp.757–772 (2016).

[12] ランサムウエア “WannaCrypt” に関する注意喚起，JPCERT コーディネーションセンター，入手先 ⟨https://www.jpcert.or.jp/at/2017/at170020.html⟩（参照 2018-12-08）．

[13] Microsoft：Windows 7 新機能　簡単「Aero スナップ」操作，Microsoft atLife，入手先 ⟨https://www.microsoft.com/ja-jp/atlife/tips/archive/windows/aero.aspx⟩（参照 2018-11-19）．

[14] OpenCV library, available from ⟨https://opencv.org/⟩ (accessed 2018-10-31).

[15] unoconv: unoconv, available from ⟨https://github.com/unoconv/unoconv⟩ (accessed 2019-03-23).

[16] ImageMagick, available from ⟨https://www.imagemagick.org/⟩(accessed 2019-03-23).

[17] zdenop: Tesseract Open Source OCR Engine, available from ⟨https://github.com/tesseract-ocr/tesseract⟩ (accessed 2017-04-17).

[18] Wu, S., Manber, U., Myers, G. and Miller W.: An O(NP) Sequence Comparison Algorithm, *Information Processing Letters*, Vol.35, No.6, pp.317–323 (1990).

[19] Winkler, W.E.: String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage, *Proc. Section on Survey Research Methods*, American Statistical Association, pp.354–359 (1990).

[20] Antoniewicz, B.: Windows DLL Injection Basics, Open Security Research, available from ⟨http://blog.opensecurityresearch.com/2013/01/windows-dll-injection-basics.html⟩ (accessed 2018-11-19).

[21] Check Point ThreatCloud Cyber Threat Map, available from ⟨https://threatmap.checkpoint.com/ThreatPortal/livemap.html⟩ (accessed 2019-03-23).

[22] Hybrid Analysis, available from ⟨https://www.hybrid-analysis.com/⟩ (accessed 2018-11-12).

[23] Pixabay, available from ⟨https://pixabay.com/ja/⟩ (accessed 2018-11-12).

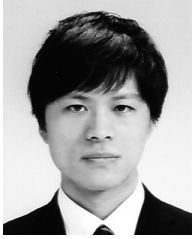
[24] 神奈川工科大学情報学部情報工学科信号処理応用研究室：標準画像／サンプルデータ，入手先 ⟨http://www.ess.ic.kanagawa-it.ac.jp/app_images_j.html⟩（参照 2018-11-12）．

[25] BLIND TEXT GENERATOR, available from ⟨http://www.blindtextgenerator.com/⟩ (accessed 2017-04-17).

© 2019 Information Processing Society of Japan

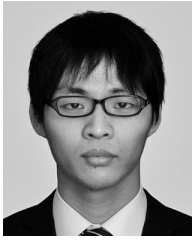
本多 俊貴

2017 年静岡大学情報学部情報科学科卒業．2019 年同大学院修士課程修了．在学中，ランサムウェア検知に関する研究に従事．



向山 浩平

2016 年静岡大学情報学部情報科学科卒業．2018 年同大学院修士課程修了．同年東海旅客鉄道株式会社入社．在学中，ユーザビリティと Web セキュリティに関する研究に従事．



白井 丈晴

2015 年静岡大学情報学部情報科学科卒業．2017 年同大学院修士課程修了．同年 KDDI 株式会社入社．在学中，ユーザ心理と QoE に関する研究に従事．

大木 哲史　（正会員）

2002 年早稲田大学理工学部電子情報通信学科卒業．2004 年同大学院理工学研究科電子・情報通信学専攻修士課程修了．2010 年早稲田大学理工学術院情報・ネットワーク専攻博士（工学）取得．2010 年早稲田大学理工学総合研究所次席研究員，2013 年産業技術総合研究所特別研究員を経て，2017 年より静岡大学大学院総合科学技術研究科講師．情報セキュリティ全般，特に個人認証を中心としたネットワークセキュリティに関する研究に従事．電子情報通信学会会員．

西垣 正勝　（正会員）

1990 年静岡大学工学部光電機械工学科卒業．1995 年同大学院博士課程修了．日本学術振興会特別研究員（PD）を経て，1996 年静岡大学情報学部助手．同講師，助教授の後，2010 年より同創造科学技術大学院教授．博士（工学）．情報セキュリティ全般，特にヒューマニクスセキュリティ，メディアセキュリティ，ネットワークセキュリティ等に関する研究に従事．2013～2014 年情報処理学会コンピュータセキュリティ研究会主査．2015～2016 年電子情報通信学会バイオメトリクス研究専門委員会委員長．2016 年より日本セキュリティマネジメント学会編集部会長．2019 年より情報処理学会情報環境領域委員長．本会フェロー．

© 2019 Information Processing Society of Japan