Vol.2017-CSEC-77 No.14
Vol.2017-IOT-37 No.14
2017/5/26

ユーザの文書編集操作を考慮したランサムウェア検知に関する一検討

本多俊貴[†1]　向山浩平[†1]　白井丈晴[†1]　西垣正勝[†2]

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．従来研究では，ファイル読み書き時のデータの特徴を用いた検知手法や，暗号化 API の利用に着目した検知手法が提案されている．前者の手法は解析環境上での検知を想定しており，後者の手法は API を使用せずに暗号化を行うことで検知が回避され得る．一般ユーザにとっては，①通常環境上で検知・実行防止を行うことが可能，②検知の回避が困難である，の 2 点を共に満たす検知方式が有用である．そこで著者らは，人間によるファイル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，①と②を共に満たすランサムウェアの検知手法を提案する．本稿では，その第一歩として，ファイル操作時におけるファイル内容表示の有無によってランサムウェアを検知・実行防止する手法を提案し，有効性評価を行った．

キーワード：ランサムウェア，ファイル保護，文書編集操作，人間らしさ

A Study on Ransomware Detection Considering User's Document Editing Operation

TOSHIKI HONDA[†1] KOHEI MUKAIYAMA[†1]
TAKEHARU SHIRAI[†1] MASAKATSU NISHIGAKI[†2]

1. はじめに

ランサムウェアによる被害が世界中で発生している．近年では特に，ファイルを不正に暗号化し，復号と引き換えに金銭を要求する暗号化型ランサムウェアの被害が顕著であり，被害者が実際に金銭を支払った事例も発生している[1][2]．そのため，暗号化型ランサムウェアに対して対策を講じる必要がある．既存研究では，検知が回避される可能性があったり，ユーザ環境でのリアルタイムな検知ができないといった課題がある．本研究では，既存研究の課題を克服するために，人間とランサムウェアによる文書編集操作の特徴の差異に着目する．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ことから，人間の通常の文書編集操作とは異なる挙動が現れる．本稿では，その第一歩として，暗号化型ランサムウェアが「ユーザに隠れてファイルを暗号化する」点に注目し，文書編集操作時におけるファイル内容の表示の有無によってランサムウェアを検知する手法を検討する．

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，感染したコンピュータ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タイプのランサムウェアである．この場合，感染したコンピュータの 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，コンピュータ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タイプの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メール添付型マルウェアなどを利用し，ランサムウェアを被害者のコンピュータに感染させる．
2. 感染後，ユーザに見えないようバックグラウンドでコンピュータ内のファイルを次々に暗号化する．
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．

画面ロック型とは異なり，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

†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
†2 静岡大学創造科学技術大学院
Graduate School of Science and Technology, Shizuoka University

る. 各セキュリティベンダや The No More Ransom Project[6]によって暗号化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，ランサムウェアの中には, シャドウコピーを格納する VSS ファイルを削除することによって，コンピュータを感染前の状態に復元することを不可能にするものも存在する[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り，近年の被害が顕著である．したがって，本稿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 暗号化に関する API コールの監視

暗号化型ランサムウェアの暗号化処理時に発生する API コールに着目して検知をする手法が重田らによって提案されている[8]．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．この手法では，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で検知の回避が可能である．

3.2 UNVEIL

暗号化型ランサムウェアと画面ロック型ランサムウェアの両方を対象とした検知方式として，Kharraz らの UNVEIL がある[9]．UNVEIL は，マルウェアのファイルアクセス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．この解析環境では，ファイルシステムへの I/O アクセスをフックすることで解析対象のマルウェアのファイルに対する動作を監視する．次に，解析対象のマルウェア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の著しい上昇」,「高いエントロピをもつ新規ファイルの生成」を確認する．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. しかしこの手法は, 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ることにより復元性をも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．また，この手法は解析環境上の検知手法として提案されているため，ユーザの利用環境においてマルウェアを検知し, 被害を未然に防ぐ手法が必要である．

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 節で述べた通り，近年被害が顕著であり，感染後にファイルの復元がより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. 3 章で述べたように, 既存研究には, 検知回避の方法が存在するという問題と，マルウェア解析環境における検知手法でありユーザ実行環境でのリアルタイム検知ができないという問題がある．これらの問題の改善を目指し，本稿では，人間の文書編集操作の特徴をホワイトリストとして用いる暗号化型ランサムウェア検知方式について検討する．

検知回避の問題については，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が図られる. 既存研究では, 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が採用されていた．この場合マルウェアは，ブラックリストから外れる新攻撃手法を何か 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新攻撃手法はすべて検知されるため，ランサムウェアが採り得る検知回避手段が大幅に制限されることになる．

リアルタイム性の問題については，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手法が必要である．そこで本稿では，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．人間の文書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」

ユーザが文書ファイルを編集する場合，画面に文書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，表示された文書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文章ファイルを編集している様子を示す．ユーザによる文書ファイル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．本稿では，これを「人間らしさ」を表す特徴として定義する．

これに対し，暗号化型ランサムウェアは，文書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内容が画面上に表示されることがない．本稿では，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，文書ファイル編集時に画面（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．

図 1 ユーザによる文書編集時の画面例

4.3 検知手法

4.3.1 人間の文書編集操作の特徴

本稿では，人間の文書編集操作の特徴を以下の 2 点として定義する．4.3.2 節で後述する検知タイミングにおいて，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) 画面上に文書編集用ウィンドウが，「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（の文書編集用ウィンドウ内）に文書ファイルの内容がされている．

(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[10]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）．本稿では，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，ユーザが文書編集作業する際の最小サイズとして設定した．

図 2 Aero スナップの最小ウィンドウサイズ（右上部）

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

WYSIWYG 型文書ファイル：

WYSIWYG（What You See Is What You Get）型の文書ファイルでは，フォント情報やレイアウト情報等がファイルに含まれており，文書編集ソフトウェアがそれらの情報をもちいてファイルの内容を表示する．Microsoft Word の文書ファイルがその典型例である．WYSIWYG 型文書ファイルは，ファイル内の情報から文書画面をコンピュータ内に再構築できるので，「ファイル編集時の実際の画面」と「コンピュータ内に再構築した文書画面」とを画像マッチングによって比較することによって，(b) の「画面上に文書ファイルの内容が表示されている」か否かの検査が可能である．

非 WYSIWYG 型文書ファイル：

WYSIWYG 型でない文書ファイルでは，フォント情報やレイアウト情報等がファイルに含まれておらず，文書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」等）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．非 WYSIWYG 型文書ファイルの場合は，ファイル内の情報から文書に含まれる文字コードが分かるので，「ファイル編集時の実際の画面から抽出された文字列」と「ファイル内に記録されている文字列」を比較することによって，(b) の「画面上に文書ファイルの内容が表示されている」か否かを検査する．

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 I/O 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[9]．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上で，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上で，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上で，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．

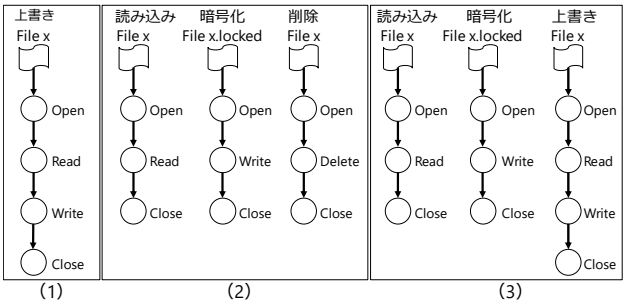

図 3 ランサムウェアのファイル I/O アクセスパターン

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」のファイル I/O アクセスが発生した時点で 4.3.1 節の(a)および(b)を検査し，人間による文書編集操作からの逸脱が検知された際には，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 節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の特定の動作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達成可能である．

5. 提案方式の実装

提案方式の第一歩として，本稿では非 WYSIWYG 型の文書ファイルの 1 つであるテキストファイル（.txt）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．

5.1 テキストファイルの内容の画面表示判定

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．

5.1.1 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．本稿では，これに OCR（Optical Character Recognition）[11]を用いる．ただし，OCR によって取得する文字列は正確ではなく，表示されている文字とは異なる文字が取得されたり，ファイルの内容以外の文字列（例えば，文書が表示されているウィンドウ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．

5.1.2 取得した文字列とファイル内容の比較

通常（短文の文書でない限り），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．本稿では，これに Diff の LCS（Longest Common Subsequence）[12]を用いる．図 4 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．

5.1.3 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（5.1.1 節）と LCS を用いて発見したファイル内の文字列（5.1.2 節）の類似度の計算には，文字列同士の近さを 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[13]．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 4 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，任意のいくつかの文書ファイルをメモ帳（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，任意のいくつかの文書ファイルをメモ帳で読み込み，当該ファイルとは異なる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．

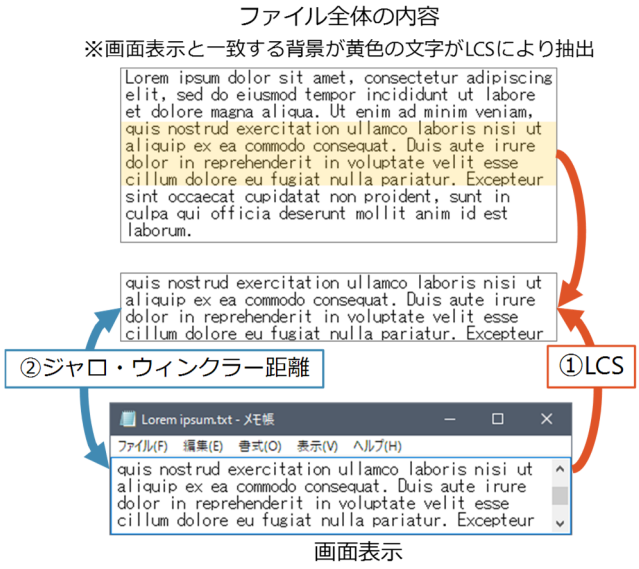

図 4 テキストファイルの内容の画面表示判定の例

5.2 検知プログラム

検知プログラムは，OS に常駐してファイル I/O アクセスを監視し，ファイル操作を行っているソフトウェアに対し，当該ソフトウェアが暗号化型ランサムウェアであるかどうかを判定する．具体的には，「4.3.2 節に示したタイミング」で「4.3.1 節に示した特徴から外れた挙動を示すソフトウェア」がないかを「5.1 節に示した処理」を用いて検査し，該当するソフトウェアが発見された場合には，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とともに，アラートを表示する．

ファイル I/O アクセスの監視は，DLL インジェクション[14]を実行し，監視対象のソフトウェアのプロセスに対して外部から監視用 DLL プログラムを埋め込むことによって実現した．すなわち，今回の検知プログラムは，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 5）.

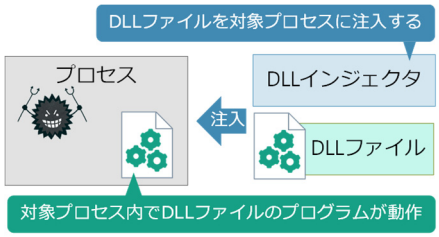

図 5 検知プログラムの構成

5.2.1 DLL インジェクタ

DLL インジェクタは他のプロセスに DLL ファイルを埋め込む．今回の DLL インジェクタは，コンピュータ上で動作しているプロセスを 0.5 秒毎に監視し，新たに起動したプロセスを発見した場合には，当該プロセスに対して DLL ファイルを注入する．（プロセス ID を入力することで，特定のプロセスのみに DLL ファイルを埋め込むことも可能である．）

5.2.2 DLL ファイル

今回の 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テキストファイルへの読み込みが発生した場合，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．

② 読み込みを実行したソフトウェアの表示ウィンドウの画面をスクリーンショットで画像として取得し，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 節で定めた「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．

③ ①で取得した文字列と②で抽出した文字列を，5.1 節で示した方法を用いて比較し，ジャロ・ウィンクラー距離を計算することによって，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．

④ ②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，③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，「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．

任意のプロセスが「ファイルの削除あるいは上書き」のファイル I/O アクセスを発行した時点で，①②③④の処理が実行されることが理想である．しかし，今回の実装においては，②③④の処理に相応の時間が要されることが判明した．このため，今回の検知プログラムでは，任意のプロセスがファイル読み込みの I/O アクセスを発生した時点で①②③④の実行を開始し，ランサムウェアが検出された際には当該プロセスのファイル削除／上書きの I/O アクセスを禁止する，という実装となっている．

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

検知プログラムを動作させ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．

6.1.2 実験方法

検知プログラムを動作させていない状態で PC が暗号化型ランサムウェア Cerber に感染した場合と，検知プログラムを動作させた状態で Cerber に感染した場合において，デスクトップ上に配置したテキストファイルの変化を確認する．Cerber は，図 3(1)の動作によってファイルを暗号化した上で，ファイル名を変更するという挙動を示す暗号化型ランサムウェアである[15]．なお，本実験は仮想 PC ソフトウェアである QEMU 上で行った．表 1 に実験環境を示す．

BLIND TEXT GENERATOR[16]を利用し，内容や文字数が様々な 40 個のテキストファイルを作成した．作成したテキストファイルの 1 つである 1000.txt の内容の一部を図 6 に示す．20 個のテキストファイルと 1 個のフォルダを仮想 PC のデスクトップ上に配置し，フォルダ内に残りの 20 個のテキストファイルを格納した．また，Prog フォルダをデスクトップに設置し，その中に検知プログラム（DLL インジェクタと DLL ファイル）を配置した．ランサムウェア動作中に検知プログラム自体が暗号化されることを防止するため，Prog フォルダ内のファイルの属性は「読み取り専用」としてある．

表 1 検知実験の動作環境

仮想マシン	QEMU	2.5.0 Debian 1:2.5+dfsg-5ubuntu10.7
	ゲスト OS	Windows 8.1
	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（ネットワークアダプタは接続）
	OS の言語設定	英語
	Windows Update	2017/2/3 17:00 までの Windows Update をすべて適用済み
	Windows Defender	無効化
検体	ファミリ	Cerber (Ransom:Win32/Cerber)
	SHA1	4a46bc5f1721d293077d611e0583d827081b6445
	ファイルパス	C:¥Users¥Owner¥Downloads¥1.exe

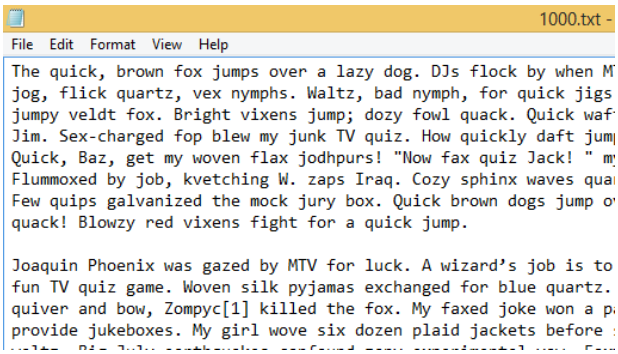

図 6 1000.txt のファイル内容の一部

6.1.3 実験結果

検知プログラムを動作させていない状態で PC にランサムウェアを感染させたところ，デスクトップならびにフォルダ内の全 40 個のテキストファイルのうち，1000.txt を含む 2 つのテキストファイルのファイル名が変更された．また，デスクトップ画面の壁紙が変更され，脅迫文が表示された（図 7）．リネームされた 2 つのファイルをメモ帳（notepad.exe）で開いて内容を確認したところ，ファイルの先頭部分はランサムウェアを動作させる前のものと同様であるが，それ以降の部分が暗号化されていた．1000.txt がリネームされたファイルである irMqkZ8Gqg.a49e をメモ帳で開いた際の画面を図 8 に示す．図 6 と図 8 を比較すると，1000.txt の内容が暗号化されてしまったことが確認できる．

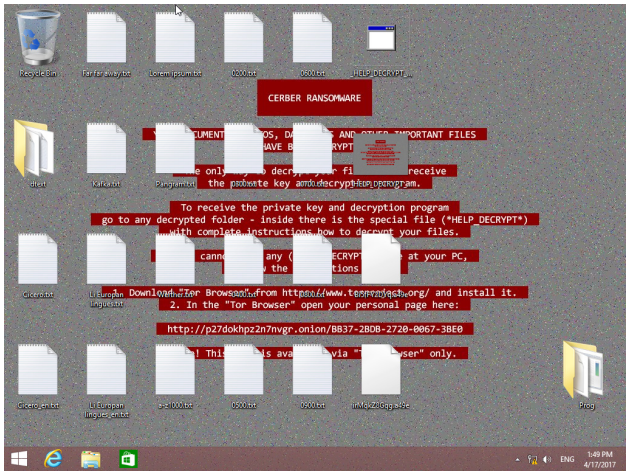

図 7 ランサムウェア感染後のデスクトップ

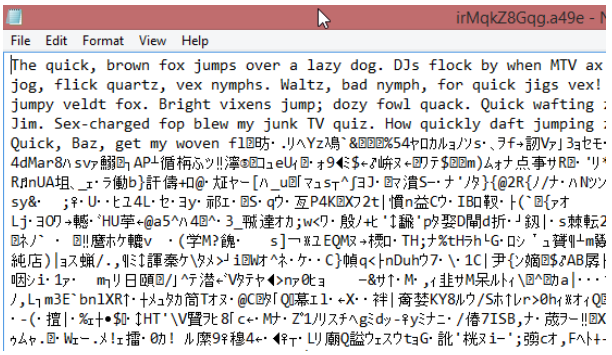

図 8 irMqkZ8Gqg.a49e のファイル内容の一部

検知プログラムを動作させた状態で PC をランサムウェアに感染させたところ，ランサムウェアを検知したことを通知するアラートが表示された．アラート表示後，1000.txt を含む 2 つのテキストファイルがリネームされ，画面に脅迫文が表示された．しかし，リネームされた 2 つのファイルをメモ帳で開いて内容を確認したところ，2 つのファイルはいずれもランサムウェアが動作する前のファイル内容と同一のものであることを確認することができた．1000.txt がリネームされたファイルである 5rpXUFRCSY.a49e をメモ帳で開いた際の画面を図 9 に示す．図 6 と図 9 を比較すると，1000.txt の内容は暗号化されておらず，ファイルの内容の保全が達成されていることが確認できる．

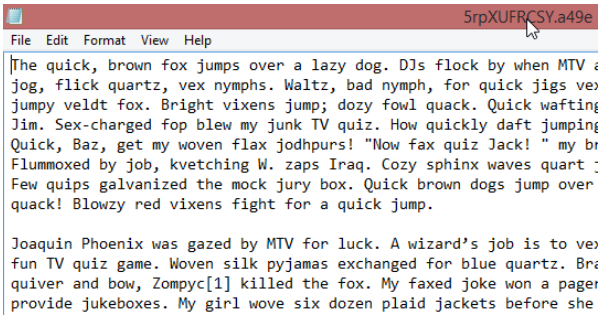

図 9 5rpXUFRCSY.a49e のファイル内容の一部

以上の結果より，ファイル名は変更されてしまったものの，検知プログラムによって暗号化型ランサムウェアの検知と暗号化の防止ができていることがわかった．提案方式のランサムウェアに対する有効性を確認することができた．

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状態で，正規ユーザが任意のテキスト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

2～10,000 文字の様々な文字列長のファイルを複数用意し，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．メモ帳のウィンドウサイズは，画面サイズの 1/4（4.3.1 節で示した最小サイズ）に設定した．その後，文字列の先頭にアルファベット「A」を追加し，約 3 秒後に上書き保存を行う．なお，ファイルの内容は BLIND TEXT GENERATOR[16]の「Pangram」ダミーテキストを利用した．表 2 に実験環境を示す．

表 2 誤検知実験の動作環境

マシン	OS	Windows 10
	画面解像度	1,920×1,080
notepad.exe	バージョン	1607
	「右端で折り返す」	有効
	ウィンドウサイズ	929×540

6.2.3 実験結果

ファイルの文字数と，検知プログラムによるジャロ・ウィンクラー距離，上書きの許可／拒否を表 3 にまとめた．

表中の「○」は上書き許可,「×」は上書き拒否を表す.「×」は，ランサムウェアとして誤検知されたことを示す．表 3 より，ファイルの文字数が少ない場合にジャロ・ウィンクラー距離が 0 となり，書き込みが拒否されていることがわかる．これは，文字数が少ない場合や，ウィンドウサイズに対して文字が表示されている面積が小さい場合に，OCR [11]が文字を認識しないことが原因であった．一方，ファイルの文字数が 10 文字以上の場合には，OCR が文字を認識し，ジャロ・ウィンクラー距離は閾値 0.7 を常に上回ることが確認できた．今回の実験結果からは，画面上に表示される文字列数が十分長い場合には誤検知は発生しないことが期待される．

表 3 ファイルごとの書き込み判定

ファイル文字数	ジャロ・ウィンクラー距離	上書き
2	0	×
4	0	×
6	0	×
8	0	×
10	0.95	○
12	0.942857	○
14	0.9625	○
16	0.936842	○
18	0.966667	○
20	0.924297	○
30	0.948822	○
40	0.961282	○
50	0.919744	○
60	0.920154	○
70	0.924384	○
80	0.901148	○
90	0.896297	○
100	0.878735	○
200	0.884622	○
300	0.821432	○
400	0.819755	○
500	0.82367	○
600	0.820362	○
700	0.815341	○
800	0.816249	○
900	0.819649	○
10000	0.826281	○

7. 考察

7.1 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一定時間以上，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悟られてしまうリスクを招くため，提案方式は，ランサムウェアを「あちらを立てれば，こちらが立たず」という状況に追い込むことができる．ここで，ファイルの内容を画面上に表示させる時間をより長く設定するほど，ランサムウェアがユーザに発見されてしまう確率が高まる．

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 節で最小ウィンドウサイズを設定している．最小ウィンドウサイズ以上の文書編集ウィンドウを表示しないソフトウェアによるファイルの上書きを拒否することで，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が困難になると期待される．

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，別の文書ファイルが画面に表示されている場合には，ファイル内容の文字列と画面表示されている文字列の類似性が低くなる（図 10）．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と考えられる．

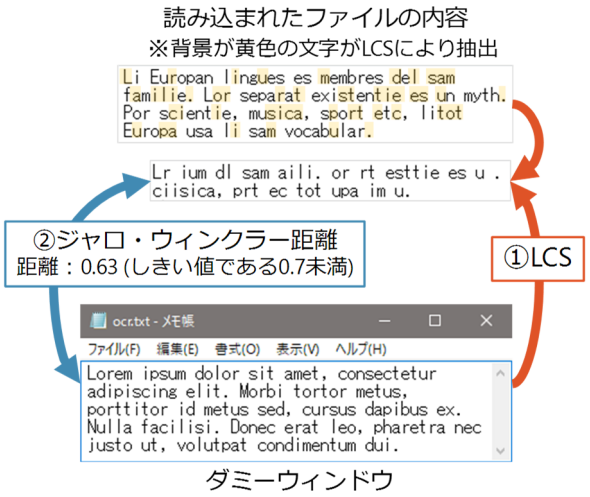

図 10 ダミー表示時の文字列の類似性検査の様子

7.2 検知プログラムの実装

本稿では，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．しかし，商用ソフトウェアの中には，DLL インジェクションを検知し，動作を停止するものも存在する．そのようなプログラムには，今回開発したランサムウェア検知用の DLL ファイルを注入することができないため，当該プログラムにランサムウェアが感染した場合には，検知ができない．この問題に対処するためには，本検知手法を OS の機能として実装し，すべてのプロセスにおけるファイル I/O アクセスと画面表示を監視することによって，

提案方式を稼働させることが必要である.

7.3 文書編集ソフト以外のファイル操作

提案方式は，ユーザによる文書編集操作を「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ため，一般的な暗号化ソフトウェアやファイル変換ソフトウェアのような「ファイルの内容を表示せずにファイル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．これらのソフトウェアは，ファイルの内容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）も発生するであろう．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が可能であると考えている.

一方で，提案方式は，文書編集操作に限らず，ユーザによるコンテンツ編集操作が発生するソフトウェア（例：画像エディタ，エクセル，パワーポイント等）に対しては，同様の方法によってランサムウェアからの保護が可能であると考えらえる.

8. まとめ

本稿では，近年その被害が顕著である暗号化型ランサムウェアに対し，「人間らしさ」と「ランサムウェアらしさ」の差異に着目してホワイトリスト型のランサムウェア検知を行い，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手法を提案した.その第一歩として，非 WYSIWYG 型文書ファイルの保護を対象としたランサムウェア検知プログラムを実装し，有効性検証を行った．有効性検証では，検知プログラムがランサムウェアの実行を検知し，文書ファイルの暗号化を未然に防止できることを検証した．また，十分な長さの文字列が画面上に表示されている場合においては，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.

今後は，他ファミリのランサムウェアに対する提案方式の有効性の確認，WYSIWYG 型文書ファイルに対応した検知プログラムの実装，ファイル内容が画面に表示されないソフトウェアを誤検知する問題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頂いた.謹んで感謝の意を表する.

参考文献

[1] 鈴木聖子：ランサムウェアに感染した病院，身代金要求に応じる，ITmedia エンタープライズ，入手先〈http://www.itmedia.co.jp/enterprise/articles/1602/19/news063.html〉（参照 2017-03-30）.

[2] CBC News：University of Calgary paid $20K in ransomware attack, 入手先〈http://www.cbc.ca/news/canada/calgary/university-calgary-ransomware-cyberattack-1.3620979〉（参照 2017-03-30）.

[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，入手先〈https://www.nomoreransom.org/〉（参照 2017-04-17）.

[7] Paul Ducklin：Got ransomware? What are your options?，Naked Security，入手先〈https://nakedsecurity.sophos.com/2016/03/03/got-ransomware-what-are-your-options/〉（参照 2017-03-30）.

[8] 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.

[9] 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).

[10] 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/tips/250.aspx〉（参照 2017-04-17）.

[11] zdenop：Tesseract Open Source OCR Engine，入手先〈https://github.com/tesseract-ocr/tesseract〉（参照 2017-04-17）.

[12] 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).

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

[14] Brad Antoniewicz : Windows DLL Injection Basics，Open Security Research，入手先〈http://blog.opensecurityresearch.com/2013/01/windows-dll-injection-basics.html〉（参照 2017-04-17）.

[15] hasherezade：Cerber Ransomware – New, But Mature, Malwarebytes Labs，入手先〈https://blog.malwarebytes.com/threat-analysis/2016/03/cerber-ransomware-new-but-mature/〉（参照 2017-04-03）.

[16] BLIND TEXT GENERATOR，入手先〈http://www.blindtextgenerator.com/〉（参照 2017-04-17）.