Computer Security Symposium 2020
26 - 29 October 2020

ランサムウェアに対する一対策とその実装

古門 良介[1,a)]　池上 雅人[2]　住田 裕輔[2]　木谷 浩[2]　白石 善明[1]　森井 昌克[1]

概要：近年, ランサムウェアと呼ばれる, サーバやパソコンを含む端末デバイス上のデータを暗号化し, その復号を条件に身代金を要求するマルウェアが流行している．典型的なランサムウェアは暗号化に際して，安全性が証明された外部の暗号ライブラリ等を利用し，公開鍵暗号方式と共通鍵暗号を組み合わせたハイブリット暗号方式で強固にデータを暗号化する．そのため，ランサムウェア感染後に暗号化されたデータを復号するのは一般に困難である．本研究では，ランサムウェアの感染時にその暗号化プロセスを乗っ取り (フッキングし), ランサムに生成される暗号鍵を意図的に任意の秘匿鍵にすり替え，暗号化データの復号を容易にする手法について提案する．また，前述の手法の実装とその評価についておこなった．

キーワード：マルウェア，ランサムウェア

A countermeasure against ransomware and their implementation

Ryosuke Kokado[1,a)]　Masato Ikegami[2]　Sumida Yusuke[2]　Hiroshi Kitani[2]　Shiraishi Yoshiaki[1]
Masakatu Morii[1]

Abstract: In recent years, malware called ransomware, which encrypts data on your PC, company's server, and so on until ransom is paid. In this paper, we introduce a technique to facilitate the decryption of ransomware by intentionally replacing the encryption key with an arbitrary secret key by hooking for the encryption function. We present the implementation and evaluation of the aforementioned functions.

Keywords: malware,ransomware

1. まえがき

近年, ランサムウェアと呼ばれる, サーバやパソコンを含む端末デバイス上のデータを暗号化し, その復号を条件に身代金を要求するマルウェアが流行している．

典型的なランサムウェアは暗号化に際して，安全性が証明された外部の暗号ライブラリ等を利用し，公開鍵暗号方式と共通鍵暗号を組み合わせたハイブリット暗号方式で強固にデータを暗号化する．そのため，ランサムウェア感染後に暗号化されたデータを復号するのは一般に困難である．したがって，ランサムウェア感染を防ぐ手法やその効力を弱体化させる手法がランサムウェア対策ではもっとも有効である．

本研究では，ランサムウェアの感染時にその暗号化プロセスに対して DLL インジェクションすることで，本来ランダムに生成される暗号鍵を任意の秘匿鍵にすり替える手法について提案する．この手法を用いることで，暗号化に用いられた鍵を把握することが可能なため，ランサムウェアによって暗号化されたデータを容易に復号することが可能となる．

また，前述の機能について実装し，3 種のランサムウェアを対象にその評価をおこなった．そして，Windows の標準 API を使用して暗号化するランサムウェア 2 種に対して，暗号鍵を任意の暗号鍵とすり替えることができ，復号が容易にできることを確認した．

[1] 神戸大学院工学研究科
Graduate School of Engineering, Kobe University
[2] キヤノンマーケティングジャパン株式会社マルウェアラボ
Malware Laboratory, Endpoint Security Technology Development Division, Endpoint Security Planning Group, Canon Marketing Japan Inc.
[a)] kokado_r@stu.kobe-u.ac.jp

2. ランサムウェア

本章ではランサムウェアについてその動作や暗号化手法を説明する．また，ランサムウェアによる暗号化データの復号手法についての関連研究を紹介する．

2.1 ランサムウェアの動作

ランサムウェアに感染すると元のファイルが削除され暗号化されたファイルに置き換えられ，多くの場合、ファイル拡張子も変更される．そして、暗号化された各フォルダにごとに復号を条件に身代金を要求する内容の脅迫文テキストが作成される．[1]

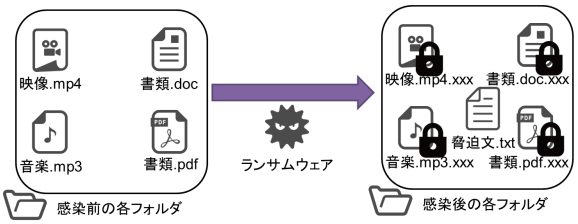

図 1 ランサムウェアの動作

2.2 一般的なランサムウェアの暗号化方法

本節ではランサムウェアが使用する暗号方式とその実装方法について説明する．

一般的なランサムウェアは暗号化に際して，安全性が証明された外部の暗号ライブラリ等を利用し，公開鍵暗号方式と共通鍵暗号を組み合わせたハイブリット暗号方式で強固にデータを暗号化する．

2.2.1 ハイブリッド暗号方式

ハイブリッド暗号方式は共通鍵暗号方式でデータを暗号化し，使用した共通鍵を公開鍵暗号方式でさらに暗号化することで安全に鍵の受け渡しを可能とする方式である．ランサムウェアではこの特徴を利用し，被害者 PC 上で暗号化に使用した鍵を攻撃者にとって安全な状態で扱っている．[2]

以下のような順序でランサムウェアはハイブリッド暗号方式を利用して，暗号化を実行する．

(1) 感染 PC で固有の鍵 K を生成，ファイル F を鍵 K で共通鍵暗号方式 (AES 等) で暗号化ファイル C_F に暗号化 E する．

$$C_F = E_K(F) \tag{1}$$

(2) K を公開鍵 P_k で暗号化 $\dot{E}$, 公開鍵で暗号化されたされた鍵 $\dot{C}_K$ を得る．

$$\dot{C}_K = \dot{E}_{P_k}(K) \tag{2}$$

(3) K を破棄する．

(4) 暗号化ファイル C_F に復号に必要な情報（$\dot{C}_K$,IV, 元ファイルサイズなど）を付け加え，ファイル F に上書きする．

(5) $\dot{C}_K$ を BASE64 符号化し脅迫文に書き込む．

また，復号は以下の順序になる．

(1) 攻撃者が攻撃者が持っている秘密鍵 P_k で，公開鍵で暗号化されたされた鍵 $\dot{C}_K$ を復号し，鍵 K を得る．

$$K = \dot{D}_{P_k}(\dot{C}_K) \tag{3}$$

(2) 鍵 K で暗号化ファイル C_F を復号し，元のファイル F を得る．

$$F = D_K(C_F) \tag{4}$$

したがって，被害者 PC 上には公開鍵で暗号化されたされた鍵 $\dot{C}_K$ のみしか存在しない．暗号化ファイル C_F の復号に必要な鍵 K は，攻撃者が攻撃者が持っている秘密鍵 P_k が必要となる．そのため，攻撃者にとって安全に被害者のファイルを暗号化が可能である．

2.2.2 ランサムウェアの暗号化処理実装

ランサムウェアに独自の暗号化処理が実装されているケースはほとんどない．独自の暗号処理を実装することは暗号化データの解析を困難にするといったメリットを得る可能性はあるが，開発コストの増加・安全強度への懸念といった多くのデメリットがある．さらに，一般に公開されている優れた処理速度で安全性が保証された暗号方式が実装された暗号ライブラリと 2.2.1 節で前述したハイブリッド暗号方式を組み合わせることで，攻撃者にとって開発コストもほとんどなく簡単に安全な暗号化処理を実装することができる．そのため，多くのランサムウェアでは Windows に内蔵されている CryptAPI や OpenSSL の暗号ライブラリを使用して，暗号化処理実装をしている．[2][3]

ここでは暗号ライブラリに CryptAPI を使用したランサムウェアの実装について説明する．図 2 に示すように，ランサムウェアは各種パラメータや暗号化対象データ CryptoAPI に渡すことで，暗号化処理を CryptoAPI を一任し，暗号化されたデータやその情報を受け取ることができる．

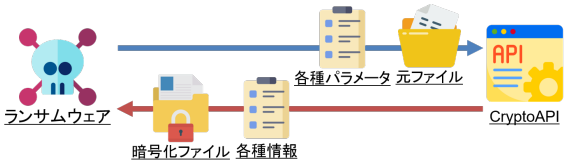

図 2 ランサムウェアの CryptoAPI の呼び出し

また，CryptoAPI 内部では図 2 に示すような関数が呼び出され処理が行われている．特に，CryptGenKey でランダムな鍵が生成され，その暗号鍵を元にファイルが暗号化されることは重要である．

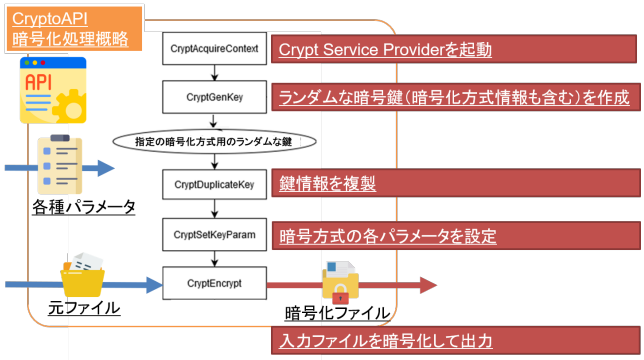

図 3 CryptAPI での暗号化処理

2.3 ランサムウェアの復号手法

ランサムウェアの復号手法は大きく 2 つに分けることができる．1 つ目は攻撃者に金銭を支払い，暗号化を解除して貰う方法である．この手法は攻撃者の目論見通りとなるが，一番確実な復号方法である．2 つ目は，攻撃者に頼らず，独自の手法で暗号鍵等を入手し復号を行う手法である．後者の手法はセキュリティ会社や有志によってツールとして提供されることもある．

独自の手法でランサムウェアの暗号化を復号するには，ランサムウェアが使用した暗号化アルゴリズム，暗号化パラメータ，ファイル暗号鍵と暗号化ファイルレイアウトの情報が必要となる．復号する上で最も重要なファイル暗号鍵の入手方法には以下の方法がある．

(1) 秘密鍵を用いて公開鍵で暗号化された鍵から復号し取得
(2) ランサムウェアの暗号化実装の脆弱性から鍵を取得
(3) 暗号化プロセスのメモリダンプから暗号鍵を取得
(4) 暗号化プロセスに介入して暗号鍵を取得・操作

1 つ目の方法は，まず攻撃者のみが知る秘密鍵を入手する必要がある．攻撃者が秘密鍵を公開するか，秘密鍵が保存された攻撃者のマシンを押収し取得するなどによって入手することができる．ハイブリッド暗号方式のランサムウェアを復号するには最も有効な方法であるが，秘密鍵が入手できることは稀な事態である．この方法は，セキュリティ会社などによるランサムウェア復号ツールにも使用されている．

2 つ目の方法は，ランサムウェアの解析をし，脆弱性を調査する必要がある．ただし，多くのランサムウェアの暗号化実装は安全性が保証されたライブラリ等を使用しており，暗号化に関する脆弱性のあるランサムウェアは稀である．この方法も，セキュリティ会社などによるランサムウェア復号ツールにも使用されている．

3 つ目の方法は，ランサムウェアの暗号時にメモリ上から暗号鍵を取得する方法である．ただし，この方法はランサムウェアがファイルごとに異なる暗号鍵で暗号化を行っていた場合や暗号方式が Black-Box 方式[*1]でない場合は効果がない．また，暗号鍵がメモリ上に存在しなければならないという制約があるので，感染後から時間が立った場合は有効でない．この方法は，古門ら [1] のランサムウェア復旧ツールで使用されている．

4 つ目の方法は，ランサムウェアの暗号化プロセスに対して，フッキングを用いて，任意の関数を実行できるような状態にし，暗号鍵を取得・操作する方法である．この方法は，Kolodenker ら [4] によって Paybreak と呼ばれるシステムとして提案された．Kolodenker らはランサムウェアの暗号化プロセスに対して，フッキングを行い，任意関数の実行を可能な状態にし，使用された暗号鍵を保存するツールを実装した．そして，VM 上で動作の確認できた 20 種類のランサムウェアのうち 12 種類に対して，暗号鍵を取得することができ，暗号化されたファイルの復号に成功したと報告した．

Paybreak システムでは，すべてのプロセスに対して，フッキングをおこない，暗号化パラメータや暗号鍵を単一のファイルに出力している．そのため，複数プロセスからの単一ファイルへの書き込みによってデッドロックが生じたり，IO アクセスが増大しシステムへの負荷が大きくなるという問題を有する．本研究では，Paybreak システムの理論を応用し，フッキングを利用した暗号化プロセスで暗号鍵をすり替える手法を提案する．Paybreak システムと比較して，OS システムへの負荷を低減させより汎用的に使用可能なツールを実装した．本研究と Paybreak システムや関連研究との相違点については 4 節で詳細を示す．

以上の 4 つの暗号鍵入手の手法は次のようにまとめられる．

表 1 暗号鍵入手手法一覧

手法番号	条件 (適用可能数)	適用タイミング
(1)	秘密鍵が入手可能（少数）	暗号化後
(2)	脆弱性がある（少数）	暗号化時・後
(3)	メモリ上に暗号鍵をロード（多数）	暗号化時
(4)	フッキングが可能（多数）	暗号化前

ランサムウェアの暗号化ファイルの復号手法にはそれぞれ制約があり，すべてのランサムウェア・状況に対して，適用可能な銀の弾丸などない．しかし，表 2.3 に示されるように手法 (4) はランサムウェア感染前から PC 上にインストールしておくだけで，多くのランサムウェアに対して有効であるため，汎用的な解決手段となる可能性がある．

また，暗号化ファイルの復号ではなく，物理ディスク上からファイルの復元を行う方法で，元ファイルを取り戻す方法がある．Wang ら [5] によって提案された TimeSSD と呼ばれる SSD のファイルシステムが提案された．この

[*1] 内部変数等が参照されることが無いという仮定のもとで安全性が保証された暗号方式

TimeSSD システムでは，ランサムウェアによるファイル削除を操作をすぐに受理し，ファイルを消去するのではなく，読み出せる形で保存しておき，復旧が可能となるファイルシステムとなっている．しかし，このシステムを使用するためには，TimeSSD システムを導入した特殊な SSD を使用する必要があるため，汎用性は高いとは言えない．

3. ランサムウェア暗号鍵すり替え手法の開発

本章では，開発で使用した関数フッキング，暗号鍵すり替え手法，すり替えた暗号鍵によるファイル復号手法，ツール仕様について説明する．

3.1 フッキング

フッキングは，あるアプリケーションに対して，元の関数の代わりに，任意関数の実行をさせることで，動作を変更するための手法である．[6]

一般的にアプリケーションの動作を変更するには，ソースコードを編集してコンパイルをし，アプリケーション自体をリビルドすることが多い．しかし，この手法は，ソースコードがない場合や，起動中のアプリケーションに対して，リアルタイムに動作を変更させることはできない．

そのため，ソースコードが入手できない上，リアルタイムに動作を変更する必要のアプリケーションに対してフッキングは非常に有効な技術である．さらに，前述のアプリケーションの特徴はマルウェアにも当てはまり，フッキングによってその動作を変えることが可能となる．

フッキングを行うためには，アプリケーションの起動時や動作時に強制的に，任意関数を読み込ませる必要があり，任意関数を DLL にコンパイルし，対象プロセスに注入する DLL injection という代表的な手法がある．また，フッキング手法の詳細は対象関数が動的リンクか静的リンクであるかに依存する．本研究では，injection 用の DLL の作成に Microsoft 社製の Detours ライブラリ [7] を使用した．Detours はフッキングに必要なアセンブラプログラミング等の低レイヤー層をラッピングし，容易にフッキングの実装を可能にする強力なライブラリである．フッキング対象は，ランサムウェアによって CryptAPI が動的にリンクされ呼び出される暗号化用の関数とした．

以下の節で，DLL injection と CryptAPI に対するフッキングについて説明する．

3.1.1 DLL injection

DLL injection は DLL を通じて対象アプリケーションのアドレス空間に任意のコードを持つ DLL を注入することである．DLL injection の実行により，対象アプリケーション上のプロセス空間で任意の関数が実行可能となる．

DLL injection は外部から対象アプリケーションに対して任意のコードを読み込ませることができる強力な手法であるため，様々なことに利用可能である．アンチマルウェアソフトからの検知を逃れるために，マルウェアが正規アプリケーション上に不正なコードを DLL injection し，正規ソフトに偽装する例も多数ある．[8] また，フッキングを実行するための足がかりとして利用する例もある．

DLL injecton の手法は様々なものがある．有名な手法の一つに，Kernel.dll へのアドレスがすべてのプロセスにおいて，共通の値をとることに着目し，Kernel.dll 中の Loadlibrary に任意の DLL を引数として与え，その Loadlibrary アドレスを引数として CreateRemoteThread を実行し，DLL injection する手法がある．対象プロセスからは，自プロセス中で Loadlibrary を呼び出し DLL を読み込んだことになるので成立する．また，Windows OS には DLL injection を行うために AppInit DLLs と呼ばれるレジストリが用意されている．[9] これは，User32.dll を読み込むすべてのプロセスの起動時に，任意の DLL を読み込ませる機能である．

本研究では，後者の AppInit DLLs を使用して，DLL injection を実装した．

3.1.2 CryptoAPI に対するフッキング

CryptoAPI は 2.2.2 節で説明した Windows で標準機能として提供されている暗号化用のライブラリである．代表的な CryptoAPI の関数を表 2 に示す．

表 2 代表的な CryptoAPI の関数

CryptoAPI 関数名	機能
CryptGenKey	ランダムな暗号鍵を作成し返す
CryptImportKey	暗号鍵をインポート
CryptEncrypt	ファイル暗号化
CryptSetKeyParam	暗号化の各種パラメータを設定
CryptDuplicateKey	暗号鍵の複製

本研究ではランサムウェアがファイル暗号化 E に使用する暗号鍵 K を任意の鍵 K_i にすり替え，復号 D を可能にした．CryptoAPI を使用するランサムウェアは暗号化の際，CryptGenKey 関数を呼び出し，暗号化方式に対応したランダムな鍵 K_R を取得し，ファイル暗号化 E に利用している．この CryptGenKey 関数をフッキングし，AES 暗号方式の場合にのみ任意の鍵と IV を持つ K_i を返す Fake_CryptGenKey 関数に入れ替えた．作成した Fake_CryptGenKey 関数内部では，CryptImportKey 関数を利用し，任意の鍵と IVK_i を読み込ませ，その鍵 K_i を返す実装をした．

その結果，ランサムウェアはファイル F を K_i で暗号化し E，暗号化ファイル C'_F を得る．

$$C'_F = E_{K_i}(F) \tag{5}$$

3.2 暗号化ファイル復号

フッキングをした状態で暗号化されたファイル C'_F は式 5 が示すように，既知である任意の鍵 K_i によって暗号化

される．したがって暗号化ファイル C'_F は攻撃者の持つ秘密鍵 P_k をする必要はなく，式 6 に示すように元ファイル F を復元可能である．

$$F = D_{K_i}(C'_F) \tag{6}$$

考慮すべき点として，ランサムウェアが使用した AES 暗号方式の暗号化モードやパディングについてはわからないので総当りで復号する必要がある．しかし，CryptoAPI で使用可能な暗号化モードは，CBC,CFB,ECB,CTS の 4 つのみである．[10] また，パディング方式は PKCS 5 のみ使用可能である．したがって，4 パターンしか組み合わせがないため，総当りで容易に使用した暗号化モードを特定可能である．

本研究では，使用した任意の鍵と IV，暗号化されたファイルを入力すると総当りで暗号化モードを特定し，復号するスクリプトを実装した．

3.3 ツール仕様

開発したツール仕様は以下のようになる．

- 対応 OS: Windows7∼Windows 10
- 対応ランサムウェア: CryptoAPI を使用して AES で暗号化を実行し，フッキング可能な検体
- DLL injection: AppInit DLLs を利用して，原則全てのプロセスに injection

本ツールは，ランサムウェアをフッキングする必要があるため，感染後には有効ではない．感染前から予め本ツールをセットアップしておく必要がある．また，ツールの導入後，ランサムウェアによって暗号化されたファイルは 3.2 節で作成したスクリプトを使用して復号する必要がある．

4. ツール評価

本章では開発したツールについて，実際のランサムウェアに対して有効かを検証し確認した．また，評価結果をもとに Paybreak システムとの相違点や既存のランサムウェア対策との相違点について説明する．

4.1 ツール評価手法

まず，表 3 に示す実験環境を VM 上に構築し，本ツールをセットアップした．

表 3 実験環境

VM ソフト	VirutalBox 6.0.12
ゲスト OS	Windows7SP1x86
ゲスト RAM	2GB

そして，表 4 に示す検体を実験環境に感染させ，ツールの動作を確認し，ファイルの復号が可能か確認した．

表 4 使用検体

検体ファミリ名	使用検体 MD5
Medusa Locker	129d3661a7341d3b069868a43714b42
Jaff Locker	a88358eb5e1efc92c74b35850ab6f2af
Cerber	b188a7a9de9c101aed6ecf075daf19f2

4.2 ツール評価結果

ツール評価結果は以下の表 5 に示すようになった．

表 5 ツール評価結果

検体ファミリ名	鍵すり替え結果	ファイル復号結果
Medusa Locker	○	○
Jaff Locker	○	○
Cerber	×	×

Medusa Locker と Jaff Locker は CryotoAPI を使用して，AES 暗号方式で暗号化するため本ツールで鍵をすり替えることができた．そして，ファイルの復号も可能であった．しかし，Cerber は CryptoAPI を使用するが，RC4 方式で暗号化するため，本ツールでは鍵をすり替えることができなかった．

ランサムウェアは一般的に AES 暗号方式を暗号化に利用するため，ツールでは AES の暗号鍵すり替え機能のみ実装していた．しかし，結果から他暗号方式についても暗号鍵がすり替える機能の実装が必要だと考えられる．

4.3 Paybreak システムとの相違点

Paybreak システムと本ツールはどちらも暗号化動作をフッキングする点では共通している．しかし，ランサムウェアのファイル暗号鍵の取り扱い方の点については大きく異なる．

Paybreak システムでは，CryptSetKeyParam 関数や CryptEncrypt 関数など 10 個以上，CryptoAPI の関数に対してフッキングし，さらにそれらの引数や暗号鍵，パラメータを単一ファイルに書き出す実装がされていた．そのため，フッキング回数が増えシステムへの負荷が大きくなりやすかった．また，単一のファイルに鍵情報を書き込む実装のため，複数プロセスからの書き込む際競合やデッドロックを引き起こし，Paybreak システムを検証した VM のディスク I/O 負荷が増大してクラッシュしてしまうことが多数あった．さらに，一般のアプリケーションに対しても CryptoAPI を利用する場合，Paybreak システムはそれらの引数や暗号鍵，パラメータを単一ファイルに書き出し，パフォーマンスに悪影響を与えていた．

本ツールでは，CryptGenKey 関数のみをフッキングするので，フッキング回数が少ないため，システムへの負荷が増大しにくい．また，ファイルに鍵情報を書き出すのではなく，あらかじめ任意の鍵を決定しておいて鍵をすり替えるのでメモリへのアクセスとなり，ディスク I/O はフッキング有無でほとんど変化しないため，本ツールを検証

した VM がクラッシュすることはなかった．さらに，本ツールは CryptGenKey 関数のみをフッキングするので，CryptoAPI を利用する一般のアプリケーションであってもランダムな鍵生成をしない場合，フッキングはされないためパフォーマンスへの影響は小さいと考えられる．

4.4 既存のランサムウェア対策との相違点

既存のランサムウェア対策と本ツールの対策ではランサムウェア自体の動作を止めるかそうでないかで大きく異る．

既存のランサムウェア対策ではリアルタイムに対象 PC をモニタリングし，既知のランサムウェアに対してはシグネチャマッチングで検知しその動作を止める．また，未知のランサムウェアに対してはその振る舞い等で検知して動作を止めるが，検知までのラグや False Negative によってファイルが暗号化されてしまうケースが発生する．

本ツールの対策では，リアルタイムにすべてのプロセスに対して，DLL injection をしてフッキングをおこなうことで，ランダムに生成される AES の暗号鍵 K_R を任意の鍵 K_i にすり替えている．そのため，検知までのラグは生じない．また，既知のランサムウェアに対してだけでなく，未知のランサムウェアであっても CryptoAPI を使用し，AES 暗号方式であれば鍵をすり替えることができるため，検知漏れリスクは低い．

5. あとがき

本稿では，強固なハイブリッド暗号方式でデータを暗号化するランサムウェアに対して，それらのファイル暗号鍵を任意の鍵ですり替え，復号を可能にする手法について提案し，実証を行い，その方法が有効であることを確認した．また，既存のランサムウェア対策とはことなり，本手法は検知漏れリスクが低く，検知ラグが生じないため，ランサムウェアによる暗号化をより無効化することが可能である．

謝辞

本研究は JSPS 科研費 20K11810 の助成を受けたものです．

参考文献

[1] 古門良介，池上雅人，長谷川智久，原田隆史，　木谷浩，森井昌克：ランサムウェア感染時の復旧対策ツールの開発【続報】，技術報告 10 (2020).

[2] Kok, S., Abdullah, A., Jhanjhi, N. and Supramaniam, M.: Ransomware, threat and detection techniques: A review, *Int. J. Comput. Sci. Netw. Secur*, Vol. 19, No. 2, pp. 136–146 (2019).

[3] 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).

[4] Kolodenker, E., Koch, W., Stringhini, G. and Egele, M.: PayBreak: Defense Against Cryptographic Ransomware, *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17 13, New York, NY, USA, Association for Computing Machinery, p. 599–611 (online), DOI: 10.1145/3052973.3053035 (2017).

[5] Wang, X., Yuan, Y., Zhou, Y., Coats, C. C. and Huang, J.: Project almanac: A time-traveling solid-state drive, *Proceedings of the Fourteenth EuroSys Conference 2019*, pp. 1–16 (2019).

[6] Berdajs, J. and Bosnić, Z.: Extending applications using an advanced approach to DLL injection and API hooking, *Software: Practice and Experience*, Vol. 40, No. 7, pp. 567–584 (2010).

[7] Microsoft: Microsoft Research Detours Package, `https://github.com/microsoft/Detours` (2020).

[8] Erdélyi, G.: Hide'n'seek? anatomy of stealth malware, *Proceedings of the 2004 black Hat Europe*, pp. 147–167 (2004).

[9] Microsoft: AppInit DLLs and Secure Boot, `https://docs.microsoft.com/en-us/windows/win320dlls/secure-boot-and-appinit-dlls` (2018).

[10] Microsoft: CryptGetKeyParam function, `https://docs.microsoft.com/en-us/windows/win32/api/wincrypt/nf-wincrypt-cryptgetkeyparam` (2018).