ランサムウェアにおける暗号化処理について

重田 貴成[1]　森井 昌克[1]　長谷川 智久[2]　池上 雅人[2]　石川 堤一[2]

概要：
ランサムウェアはパソコン，あるいはサーバ内のファイルをその復号鍵を知らせることなく暗号化し，ファイルの所有者が復号できなくなることによって被害を及ぼすことになる．鍵の配送には RSA 暗号が使われることが一般であり，実際のファイルの暗号化には AES が使われることが多い．AES については，独自のプログラムやツールを使うことは稀であり，既存の暗号化復号ツール，あるいはライブラリが用いられることが多い．本稿では，この AES による暗号化処理に着目し，既存のライブラリ等が意図せず利用されることを検知し，ランサムウェアによる暗号化処理が行われているか否かを検知する方法について考察する．本方法によって，ランサムウェアの処理を中断させることが可能になり，被害を最小限に食い止められる可能性がある．

キーワード：ランサムウェア, ランサムウェア対策, 暗号化処理, CryptoAPI, OpenSSL

Encryption Processing of Ransomware

Takanari Shigeta[1]　Masakatu Morii[1]　Tomohisa Hasegawa[2]　Masato Ikegami[2]
Teiichi Ishikawa[2]

Abstract: In recent years, the prevalence of ransomware is a serious problem. Ransomware encrypts user's file without permission, and demands a ransom for decryption. Ransomware uses encryption algorithms such as AES and RSA. This paper presents analysis of ransomware's encryption processing, and considers methods against ransomware.

Keywords: ransomware, anti-ransomware, encryption processing, CryptoAPI, OpenSSL

1. まえがき

近年，ランサムウェアの流行が深刻な問題となっている．ランサムウェアはパソコンやサーバ内のファイルを，復号鍵を知らせることなく暗号化し，ファイルの所有者が復号できなくなることで被害を及ぼす．復号鍵は攻撃者の用意したサーバ上に存在するため，自力での復号は難しい．そのため，暗号化が始まった後の早い段階でランサムウェアを検知し，処理を中断させることで，被害を最小限に食い止められる可能性がある．

ランサムウェアの多くは暗号化アルゴリズムに AES [1] と RSA [2]，暗号化ライブラリに CryptoAPI [3] または OpenSSL [4] を利用する．本稿では，CryptoAPI または OpenSSL を利用したランサムウェアの検知と対策を行う手法を提案する．提案手法では，まずプロセス一覧の監視を行う．そして，新しいプロセスが起動した際に，そのプロセスが CryptoAPI や OpenSSL を利用しているか確認する．CryptoAPI の検知は，メモリ上にあるインポートアドレステーブルを探索して行う．OpenSSL の検知は，AES の暗号化に使用されるテーブルをメモリ上から探索して行う．これらの暗号化ライブラリを検知したときに，暗号化処理を中断させ，ユーザが動作の継続・強制終了を選択するメッセージボックスを表示する．

[1] 神戸大学大学院工学研究科
Graduate School of Engineering, Kobe University
[2] キヤノン IT ソリューションズ株式会社
Canon IT Solutions Inc.

表 1　ランサムウェアが使用する暗号化アルゴリズムとライブラリ
Table 1　Encryption Algorithms and Library of Ransomware.

ランサムウェア	アルゴリズム	ライブラリ
Cryptographic Locker	AES	CryptoAPI
CryptoLocker	AES, RSA	CryptoAPI
CryptoWall	RSA	CryptoAPI
Locky	AES, RSA	CryptoAPI
Critoroni / CTB Locker	AES, ECDH	OpenSSL
TorrentLocker	AES	OpenSSL
Dirty Decrypt	RC4, RSA	なし

検証実験では，CryptoAPI と OpenSSL を用いて実装したプログラムに対して提案手法を適用し，実際に検知と動作選択のメッセージボックスが動くことを確認した．また，T-table の格納されている実行ファイルの領域に関しても検証を行い，rdata セクションまたは text セクションに格納されていることを確認した．

2. ランサムウェアと暗号ライブラリ

マルウェアの一種にランサムウェアがある．ランサムウェアはパソコンやサーバ内のファイルを，復号鍵を知らせることなく暗号化し，ファイルの所有者が復号できなくなることで被害を及ぼす．ランサムウェアの例として，Locky の暗号化処理を以下に示す．

(1) 攻撃者のサーバで RSA の公開鍵・秘密鍵を生成し，公開鍵を感染 PC に送信する．
(2) 感染 PC では，AES でファイルを暗号化する．
(3) AES の共通鍵を RSA の公開鍵で暗号化し，PC に保存する．

AES で暗号化されたファイルを復元するためには，AES の共通鍵が必要である．しかし，共通鍵は RSA の公開鍵で暗号化されている．よって，共通鍵を復元するためにはサーバ上にある RSA の秘密鍵が必要となるため，自力での復号は難しい．

ランサムウェアの暗号化処理の実装では，独自のプログラムを使うことは稀であり，既存のライブラリが用いられることが多い．主なランサムウェアで用いられている暗号化アルゴリズムと暗号化ライブラリを**表 1** に示す．

CryptoAPI [3] は，Microsoft Windows に組み込まれている暗号化用ライブラリである．Windows 95 から存在する標準ライブラリであるため，Windows のバージョンを気にすることなく利用できる．また，OS に標準で組み込まれているライブラリであるため，暗号化に必要な定数やコードを実行ファイルに埋め込む必要がなく，ファイルサイズを削減できる．しかし，Windows 以外では利用できないため，マルチプラットフォームのソフトウェアには向かない．

OpenSSL [4] は，オープンソースで開発されている暗号化ライブラリである．マルチプラットフォームに対応しており，Windows や Linux 上で動くさまざまなプログラムで利用できる．しかし，暗号化に必要な定数やコードをを実行ファイルに埋め込む，もしくは DLL を同封する必要があり，ファイルサイズが肥大化する．

表 1 より，ランサムウェアの暗号化に関して以下の特徴があることがわかる．

- 暗号化アルゴリズムに AES を利用することが多い
- 暗号化ライブラリに CryptoAPI と OpenSSL を利用することが多い

以上を踏まえて，本稿では以下のようなランサムウェアを検知する手法を提案する．

- 暗号ライブラリに CryptoAPI を利用しているランサムウェア
- 暗号化アルゴリズムに AES，暗号ライブラリに OpenSSL を利用しているランサムウェア

3. ランサムウェアの実行検知手法

本章では，「プロセス監視段階」と「ランサムウェアの実行監視・対策段階」の 2 段階にわけて，ランサムウェアを検知する手法を提案する．プロセス監視段階では，新しく起動したプロセス一覧を取得し，それらのプロセスに対してランサムウェア検知用のコードを注入する．ランサムウェアの実行監視・対策段階では，プロセスが CryptoAPI や OpenSSL を利用しているかどうか調べ，利用している場合にはその対策を行う．

3.1 プロセスの監視

本節では，プロセス監視段階の手法を提案する．提案手法の手順は以下のとおりである．

(1) 1 秒毎にプロセス一覧を取得し，新しく起動したプロセスを取得する．プロセス一覧の取得間隔は，以下の要因から決定した．
  - プロセス一覧の取得による平均 CPU 使用率が低い
  - SSD の書き込み速度が 250 MB/s であることを考慮すると，ランサムウェアの暗号化速度に対して十分速い

(2) 起動したプロセスのうち，ホワイトリストに含まれているものを除外する．ホワイトリストには，システムプロセスや信頼できるプログラムを入れておく．これにより，実行監視の負荷やユーザの負担を減らすことができる．

(3) 新しく起動したプロセスに対して，DLL インジェクション [5] を行う．DLL インジェクションは，他プロセスに対して DLL をロードさせ，任意のコードを実行する手法である．DLL インジェクションでは，注入する側のプロセスが LoadLibrary 関数のアドレスを取得し，その関数を注入される側のプロセスで実行させることで，任意の DLL を注入する．ただし，64 ビッ

トプロセスから 32 ビットプロセスに対して DLL インジェクションを行う場合は，以下のことに注意する必要がある．

- 32 ビットプロセスに対して 64 ビット DLL を注入することはできない．そのため，32 ビットプロセスに対しては 32 ビット DLL を注入する．
- 32 ビットプロセスの kernel32.dll と 64 ビットプロセスの kernel32.dll は異なるため，32 ビット用の LoadLibrary 関数のアドレスを 64 ビットプロセス側から取得できない．そのため，LoadLibrary 関数のアドレスを取得する際は，別途起動した 32 ビットプロセス内でアドレス取得する．

(4) DLL インジェクションが成功すると，注入した DLL のメイン関数が OS によって実行される．よって，DLL のメイン関数を通して任意のコードを実行できる．

3.2 ランサムウェアの実行監視・対策

本節では，新しく起動したプロセスに対し DLL インジェクションを実行した後，そのプロセスがマルウェアかどうかを検知する手法を提案する．提案手法の手順は以下のとおりである．

(1) OpenProcess 関数でプロセスハンドルを取得する．プロセスハンドルの実体は，メモリ上に展開された実行ファイルの先頭を指すアドレスである．

(2) メモリ上に展開された実行ファイルを解析し，CryptoAPI および T-table を検知する．解析の具体的な手法は 3.2.1 節で述べる．

(3) CryptoAPI または T-table が検知された場合は，プロセスを中断させる．

(4) プロセスを再開させるか，強制終了させるかを選択するメッセージボックスを表示し，ユーザに選択を促す．

3.2.1 暗号化ライブラリの検知

CryptoAPI の実装は，システムフォルダ中の advapi32.dll の中に存在する．DLL でエクスポートされた関数呼び出しは，インポートアドレステーブルというテーブルを介して，間接アドレス指定方式で行われる．つまり，関数呼び出し命令のオペランドには，関数実体のアドレス（インポートアドレス）が入ったアドレスが指定されている．このインポートアドレスの配列が，インポートアドレステーブルである．よって，インポートアドレステーブルを探索し，advapi32.dll 内の関数がエクスポートされていることを確認することで，CryptoAPI を検知できる．

2 章で述べたとおり，ランサムウェアは暗号化アルゴリズムに AES を使うことが多い．よって，OpenSSL に関しては AES を利用するランサムウェアに対して検知を行う．OpenSSL の AES は T-table [6] を用いて実装されている．T-table とは，暗号化処理を高速化するために，暗号化処理の一部をテーブルにまとめたものである．よって，OpenSSL を用いて AES の暗号化を行っている実行ファイルには，内部に T-table が存在する．実行ファイル内にはセクションという領域が複数存在し，格納されている情報の種類に応じて分割されている．格納されるセクションはコンパイラ依存であるが，rdata セクションには定数が，text セクションには命令が格納されることが多い．

(1) プロセスハンドルから実行ファイルを解析し，rdata セクションおよび text セクションのアドレスを取得する．

(2) rdata セクションおよび text セクションから T-table を線形探索する．

3.2.2 プロセスの停止・再開・強制終了

プロセスの停止は，メッセージボックス表示用以外のスレッドに対して SuspendThread 関数を呼ぶことで行う．プロセスの再開は，メッセージボックス表示用以外のスレッドに対して ResumeThread 関数で呼ぶことで行う．プロセスの強制終了は，TerminateProcess 関数で呼ぶことで行う．

4. 実験結果

以下の環境で CryptoAPI を利用したプログラム，および OpenSSL を静的リンクしたプログラムを作成し，暗号化ライブラリを検知できるかどうか検証実験を行った．

OS Windows 10 Home
CPU Intel(R) Core(TM) i5-4430 3.00GHz
RAM 4.00GB
コンパイラ Visual Studio 2015

実験の結果，CryptoAPI を用いたプログラムはインポートアドレステーブルに，関数実体アドレスが格納されていることを確認した．また，OpenSSL を静的リンクさせて AES の暗号化を行うプログラムは，メモリ上に展開された実行ファイル中に T-table が格納されていることを確認した．このとき，x86 コンパイルの場合は rdata セクション，x64 コンパイルの場合は text セクションに格納されていることを確認した．また，暗号化ライブラリの検知後，プロセスを停止させてメッセージボックスを表示できた．そして，ユーザの意図しない暗号化が行われている場合に，プロセスを停止させ，動作の継続・強制終了をユーザが選択できることを確認した．

5. まとめと今後の課題

本稿では，ランサムウェアの AES による暗号化処理に着目して，CryptoAPI や OpenSSL といった既存のライブラリが意図せず利用されることを検知し，ランサムウェアによる暗号化処理が行われているか否かを検知する方法について考察した．本方法によって，ランサムウェアの処理を中断させることが可能になり，被害を最小限に食い止められる可能性がある．

提案手法の問題点および今後の課題として，以下のことがあげられる．

新しいプロセスが大量に起動する場合の負荷

もし短い間隔で大量のプロセスが起動する場合，その数だけインポートアドレステーブルやセクションを探索する必要があるため，CPU の負荷が増大する．

T-table の格納されるセクション

OpenSSL の T-table が格納されるセクションは，コンパイラに依存する．そのため，Visual Studio 2015 以外でコンパイルされた OpenSSL ライブラリの場合，rdata セクションや text セクション以外に T-table が格納される可能性がある．今後，さまざまなコンパイラに対して T-table がどのセクションに格納されるか調査する必要がある．

ホワイトリストによる検知漏れ

ランサムウェア自身が DLL インジェクションを行い，ホワイトリストに含まれるプロセス上で暗号化処理を実行した場合，検知漏れがおきる．

参考文献

[1] Daemen, J. and Rijmen, V.: *The design of Rijndael: AES-the advanced encryption standard*, Springer Science & Business Media (2013).

[2] Rivest, R. L., Shamir, A. and Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, Vol. 21, No. 2, pp. 120–126 (1978).

[3] Microsoft: Cryptography Functions (Windows), `https://msdn.microsoft.com/en-us/library/windows/desktop/aa380252(v=vs.85).aspx`. (2016/08/12).

[4] The OpenSSL Project: Cryptography Functions (Windows), `https://msdn.microsoft.com/en-us/library/windows/desktop/aa380252(v=vs.85).aspx`. (2016/08/12).

[5] Richter, J.: Load your 32-bit DLL into another process space using INJLIB, *Microsoft Systems Journal* (1994).

[6] Daemen, J. and Rijmen, V.: AES proposal: Rijndael (1999).