Vol.2016-DPS-166 No.24
Vol.2016-CSEC-72 No.24
2016/3/4

マルウェアから暗号処理を特定する手法の実装と評価

西川　弘毅[†1]　山本　匠[†1]　河内　清人[†1]

昨今のマルウェアの中には，暗号技術を用いて通信を秘匿するものがある．暗号化された通信ログからでは，マルウェアにより窃取された情報の特定が困難である．そのため，マルウェアが暗号処理に用いている暗号アルゴリズムと鍵を特定し，暗号化された通信ログを復号することで，通信を明らかにする技術が必要である．そこで，印刷可能文字列が，暗号処理の入出力で比較すると入力の方が，印刷可能文字列の割合が多くなることを利用して，マルウェアを実行した際の動作ログである実行トレースから，暗号処理を自動的に特定する．本手法に対する評価のために，暗号処理を行うプログラム 2 つ用意し，暗号処理の特定について評価を行った．どちらのプログラムにおいても，暗号処理の候補を絞り込むことに成功した．

Method to Identify Encryption in Malware

Hiroki Nishikawa[†1] Takumi Yamamoto[†1] Kiyoto Kawauchi[†1]

Recently, some malwares encrypt communication to conceal what transmitted is. When the communication is encrypted, it becomes difficult to identify what malware attack is, what information is stolen from communication logs. Therefore, identifying encryption logic and key is needed to reveal the communication. Some methods to identify encryption algorithm from execution traces using features of encryption logic are proposed. In this paper, we use feature of encryption that input of encryption has many printable words than output for detect encryption automatically. We evaluated two program executing encryption. Each case, we can narrow down candidate encryption part of those programs.

1. はじめに

近年，機密情報の窃取を目的に，企業や官公庁に対して行われる標的型攻撃が多発し，セキュリティ上の重大な脅威となっている．警察庁によると，サイバーインテリジェンス情報共有ネットワークを通じて把握された標的型メール攻撃は，平成 27 年上半期には 1472 件に上り，前年同期比で 1256 件(581%)の，急激な増加となっている[1]．標的型メールの特徴としては，非公開のメールアドレスに対しての攻撃が全体の 9 割を占めることから，攻撃者が，攻撃対象の組織や職員について調査し，周到な準備を行ったうえで攻撃を実行していることが伺える．

一般的な標的型攻撃は，巧妙に文面を細工したメールを攻撃対象に送信するところから開始される．同メールにはマルウェアを含んだ文書ファイルが添付されており，メール受信者が同文書を開いた瞬間，ユーザの端末はマルウェアに感染してしまう．攻撃者は同マルウェアをインターネット上の指令サーバ(C&C サーバ: Command and Control サーバ)から制御し，標的組織内部のネットワークから機密情報を探索，アップロードすることで目的を達成する．これらの情報漏洩被害の深刻化を背景に，パソコンやサーバ等が生成したログや，マルウェアを解析することで，マルウェアの感染端末内での挙動を明らかにするネットワークフォレンジック技術が注目されている．

しかし昨今のマルウェアの中には，暗号技術を用いて通信を秘匿するものがある．このようなマルウェアの通信は，暗号化された状態で記録されるため，そのままでは解析することができない．そのため，マルウェアが通信で用いている暗号アルゴリズムと，暗号化に用いる鍵を特定して，暗号化された通信を復号することが必要となる．この作業はマルウェアのリバースエンジニアリングが必要となり，膨大な手間と時間を必要とするため，自動的にマルウェアの暗号アルゴリズムを特定できるツールが求められ，[2]-[4]において研究されている．

これらの手法は，マルウェアの検体を動作させて得られる動作履歴である実行トレースを解析することで，マルウェアの暗号アルゴリズムを特定している．しかしこれらの手法は，処理に時間がかかる，誤検知が多いといった課題がある．そこで本稿では，[5][6]で著者らが示した手法の内，印刷可能文字が，暗号処理の入出力で比較した際に，入力の方が，印刷可能文字の割合が多くなることを利用する手法の評価の結果を示す．評価対象には暗号処理を行うプログラム二つを選定し，どちらのプログラムにおいても，暗号処理の候補を絞り込むことに成功した．

本稿の構成は，2.で関連研究について示し，3.で提案手法の説明を行う．4.で評価実験について述べ，5.で実験の考察を示す．

2. 関連研究

マルウェアの検体から，マルウェアが暗号処理を行っているバイナリを特定する技術として，文献[2]-[4]が知られている．これらの手法はマルウェアを実行して得られたログである実行トレースを解析して，暗号処理を構成するバイナリを特定する．実行トレースの例を図 1 に示す．実行

[†1] 三菱電機株式会社　情報技術総合研究所
Mitsubishi Electric Corporation, Information Technology R&D Center.

ⓒ 2016 Information Processing Society of Japan

Vol.2016-DPS-166 No.24
Vol.2016-CSEC-72 No.24
2016/3/4

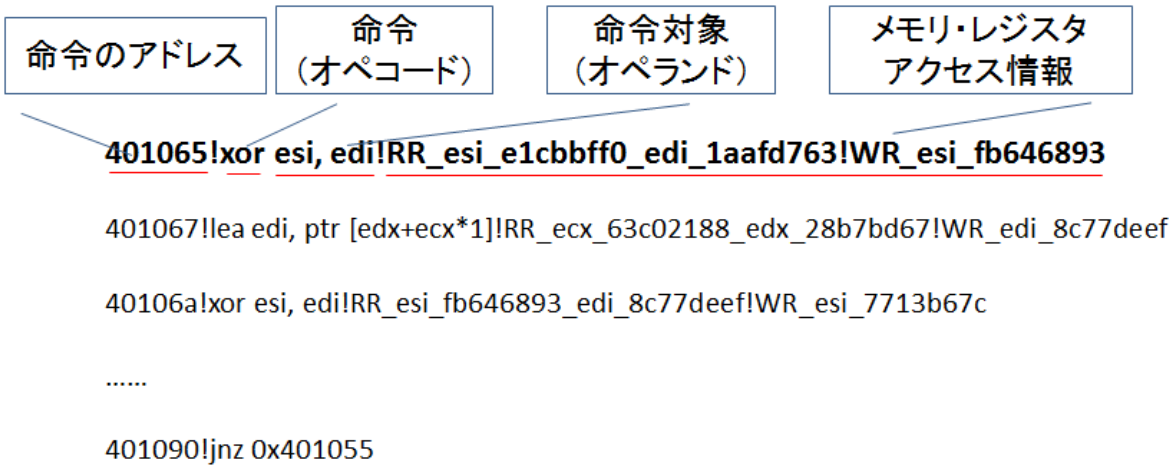


図 1　実行トレースの例

トレース一行分の構成は，命令のアドレス，命令，命令対象，メモリ・レジスタへのアクセス情報となっている．ブロックとは，プログラム中の繰り返し処理や，ジャンプ命令で構成されるループや，call，ret 命令により定義される関数などで表現される基本構造を指す．このブロックの中でも，暗号処理を行っているブロックのことを暗号ブロックと定義する．実行トレースを利用した暗号アルゴリズムの特定は，静的解析を用いたリバースエンジニアリングと比較して難読化に強い特徴があるため，難読化が施されたマルウェアへの対抗策として有効な手段といえる．実行トレースの抽出には Pin[7]等のツールを用いる．

以下に，関連研究[2]-[4]について概要と課題を説明する．

文献[2]の手法である Aligot は，マルウェアの実行トレースからループ構造に着目して暗号ブロックを抽出している．この手法におけるループ検知の特徴として，for 文などの制御命令を伴わずに，同一の命令を何度も書き込むことでループを表現する展開されたループに対しても，抽出することが可能なループ検知手法を用いることで，暗号ブロックの検知精度を高めている．また，その暗号が用いているアルゴリズムの特定では，マルウェアが用いるアルゴリズムが既知のアルゴリズムであることが多いという仮定で，暗号ブロックの候補の入出力と一致するような既知アルゴリズムの入出力を探し出し，一致した既知アルゴリズムを暗号のアルゴリズムとして判定している．しかし，この手法ではループを検知する際に，ジャンプ命令といった制御構造で表現されたループの抽出とは異なり，どの命令の連続がループとなるのかを事前に予測できず，読み込むたびにループを構成する可能性がある命令列が増加していく．そのため，ループを特定するには，ループを構成する可能性がある命令列を全て記憶し，組合せを総当たりで探索する必要があるため，処理が遅く，大量のメモリ領域を必要とするという課題があった．また，マルウェアが未知のアルゴリズムを暗号に利用している場合，検知することができないという課題があった．

文献[3]の手法である kerckhoffr は，ジャンプ命令に着目してプログラム内のループ構造を探し出し，ループを含むブロックに対して，暗号処理に特有の命令列や定数の組み合わせをシグネチャとして暗号が用いているアルゴリズムの特定を行う．シグネチャを用いた解析であるため，特定できるアルゴリズムは，Aligot と同様に既知のものに限定されてしまうという課題があった．

文献[4]の手法である ReFormat は，算術・ビット演算が暗号処理では頻繁に実行されることを利用して，実行トレースの初めからこれらの演算を数え上げていき，その時点での全体における算術・ビット演算の割合を計算し，その割合のピークによって暗号処理で用いられているブロックを特定する．しかしこの手法では，実行トレース中に複数の暗号処理で用いられるブロックが存在していると，ピークを正しく検知できず，正しく検知することができないという課題があった．

3. 提案手法

図 2 に提案手法の流れを記載する[5]．提案手法では，従来技術と同様に，まず解析対象の実行ファイルを仮想環境上で実行させ，実行トレースを取得する．2.の関連研究で述べたとおり，実行トレースは，プログラムを実行した際の実行履歴であり，一命令単位でアクセスしたレジスタやメモリの情報が記録される．

実行トレースを解析し，ループ構造や関数などのブロックを特定する．さらにブロックの入出力となるバッファを特定する．

暗号アルゴリズムによく見られる特徴や，マルウェアが暗号を利用する際の特徴に着目し，ブロックとその入出力を解析する．解析結果に基づき，暗号アルゴリズムの候補を特定する．

ⓒ 2016 Information Processing Society of Japan

Vol.2016-DPS-166 No.24
Vol.2016-CSEC-72 No.24
2016/3/4

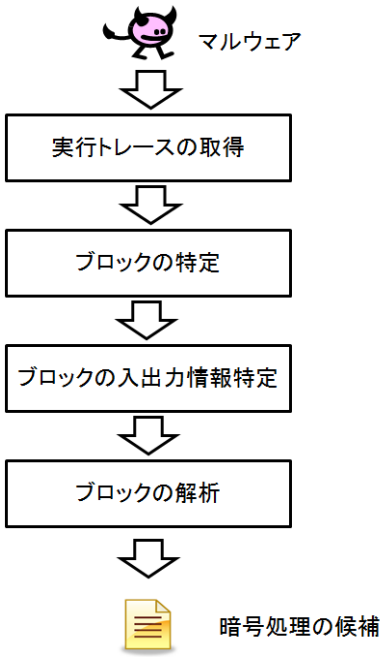


図 2　提案手法の流れ

なお提案手法は，検査対象プログラムが必ず暗号処理を実行することを前提としている．次節から，図 2 の各ステップについて詳しく説明していく．

3.1 実行トレースの取得

実行トレースの取得には，Intel が提供する Pin[7]を利用する．Pin は，プログラムの性能解析や，セキュリティ用途に用いられるツールであり，1 命令実行単位で用途に応じたコールバック関数を呼ぶことができる．コールバック関数をカスタマイズすることで，各命令がアクセスした情報について，必要なものを収集することができる．

3.2 ブロックの特定

2.の関連研究でも示したが，ブロックとは，プログラム中のループ，関数などの基本構造を指す．そのため，実行トレースを解析し，実行された命令列からブロックを特定する．

今回，実行トレース上の call 命令と ret 命令のペアを探すことで関数を検出する．call 命令が実行された後に初めて実行された ret 命令を同 call 命令のペアとなる ret 命令とする．call 命令直後のアドレスから ret 命令までの命令列をブロックのボディとする．

3.3 ブロックの入出力情報の特定

暗号処理は，入力として平文と鍵を受け取り，暗号文を出力する．本稿では暗号アルゴリズムの入出力の特徴に基づいた解析を行う．そのため，ブロックの入出力となるバッファを，Aligot と同様の手法で特定する[2].

3.3.1 入力バッファ

あるブロックの入力バッファは，以下の条件を満たすメモリとする．

- ブロックが実行される前に定義（書き込み）されるメモリ（バイト/ワード）
- ブロックの中で再定義される前に参照（読み出し）されるメモリ（バイト/ワード）

3.3.2 出力バッファ

あるブロックの出力バッファは，以下の条件を満たすメモリとする．

- ブロックの中で定義されるメモリ（バイト/ワード）

3.4 ブロックの解析

特定したブロックの入出力情報を元に，ブロックが暗号処理を行っているものかを特定する．

■　暗号処理の入出力特徴に基づく解析

暗号処理に平文を入力すると暗号文が出力される．暗号文はランダムなバイト列であるため，暗号処理の出力に，印刷可能な文字が含まれることは多くはないと考えられる．

ここで，印刷可能な文字とは，画面に出力することが可能な制御文字を除く文字のことである．

一方，マルウェアが感染端末で得た情報を暗号化して外部に送信することを想定すると，暗号処理の入力，すなわち，平文には，印刷可能文字が多く含まれていると考えられる．

そこで，ブロックの入出力の印刷可能文字の割合を比較することで，同ブロックが暗号処理の候補かどうかを判定する（図 3）．解析手順は次のとおりである．特定したブロックの集合を B とする．

① 確認済みブロック集合 UB を空にする．
② ブロック b（b∈B，b∉UB)を一つ選択する．UB に b を追加する．該当するブロックが無ければ終了する．
③ $\boldsymbol{I_b}$ をブロック b の入力バッファの集合，$\boldsymbol{O_b}$ をブロック b の出力バッファの集合とし，$\boldsymbol{I_b}$ と $\boldsymbol{O_b}$ から印刷可能文字の割合を計算し，それぞれ Ri，Ro とする．
④ Ri-Ro>θ ならば，b を暗号処理，Ri-Ro<θ ならば b を復号処理とする．ここで，θ は閾値である．
⑤ ②を実行する．

ステップ③について説明する．印刷可能文字の割合は，入力（または出力）から得られる印刷可能文字数の総和を入力（または出力）の数で割ったものである．

ⓒ 2016 Information Processing Society of Japan

Vol.2016-DPS-166 No.24
Vol.2016-CSEC-72 No.24
2016/3/4

表 1　TEA で検出された候補ブロックの情報

No.	開始アドレス(16 進)	終了アドレス(16 進)	サイズ (16 進)	実行トレース行数	Ri	Ro	Ri - Ro
1	1321c15	1321c28	13	11	0.45	0.25	0.2
2	1321310	13214dd	1cd	178	0.381	0.222	0.159
3	1321020	132109f	7f	763	0.357	0.205	0.152
4	1321310	13214dd	1cd	9	0.388	0.222	0.166
5	13210a0	1321274	1d4	1456	0.456	0.241	0.215

表 2　RC4 で検出された候補ブロックの情報

No	開始アドレス(16 進)	終了アドレス(16 進)	サイズ (16 進)	実行トレース行数	Ri	Ro	Ri - Ro
1	f823d0	f8259d	1cd	150	0.402	0.173	0.229
2	f823d0	f8259d	1cd	174	0.349	0.198	0.151
3	f823d0	f8259d	1cd	266	0.491	0.21	0.281
4	f81060	f810ce	6e	6418	0.704	0.349	0.355
5	f810d0	f8118d	bd	1015	0.5	0.336	0.164
6	f823d0	f8259d	1cd	194	0.386	0.185	0.201
7	f81190	f813a2	212	3493	0.32	0.164	0.156
8	f81420	f81673	253	12438	0.488	0.261	0.227

表 3　暗号処理の候補となるブロック数

プログラム名	候補ブロック数	総ブロック数
TEA	5	24
RC4	8	48

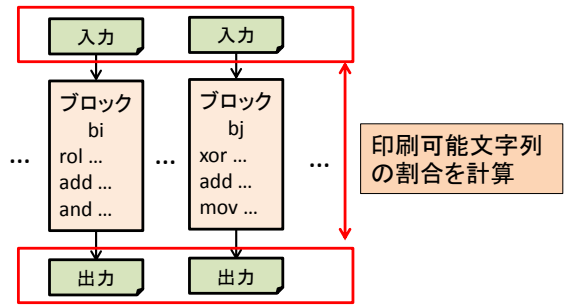


図 3　ブロックの入出力情報解析の様子

4. 評価実験

4.1 準備

提案手法の評価対象として，[8], [9]から入手した，暗号処理を行うプログラムを選択した．

このプログラムを実行すると，ハードコードされた鍵，平文を暗号化し，それぞれ画面に出力する処理を行う．その様子をそれぞれ図 4，図 5 に示す．

今回，印刷可能文字として，ASCII による判定を行った．また，暗号処理と判定する閾値 θ として，0.15 を選択した．

4.2 実験結果

今回の実験により，それぞれのプログラムから，暗号処理の候補となるブロックがいくつか得られた．

TEA と RC4 のプログラムにおいて検出された各ブロックに関して，ブロックの開始アドレスと終了アドレス，実行トレース行数，入出力の印刷可能文字の割合 Ri と Ro について，それぞれ表 1，表 2 に示す．また，表 3 に，暗号処理の候補となるブロック数を示す．

ここで，TEA と RC4 の実行ファイルに対して解析を行い，各ブロックの特徴を調査した．

4.2.1 TEA で検出されたブロックについて

TEA の実行ファイルを解析し，各ブロックについての情報を整理したものを表 4 に示す．

ここで，No2 と No4 について，この関数は三回呼ばれているにも関わらず，検出されたものは二つだった理由を調査した．検出されなかったブロックは，”Key”を出力する部分であり，入力の中に印刷可能文字が少なかったためであると考えられる．

この解析結果より，TEA の暗号処理を特定することができたと言える．

4.2.2 RC4 で検出されたブロックについて

RC4 の実行ファイルを解析し，各ブロックについての情報を整理したものを表 5 に示す．

この解析結果より，RC4 の暗号処理を特定することがで

ⓒ 2016 Information Processing Society of Japan

きたと言える．

5. 考察

5.1 誤検知と閾値について

現状，どの値が適切な閾値であるかの判断はできていない．しかし，今回は印刷可能文字を ASCII としたが，ASCII に含まれない文字を印刷可能文字とみなさず，出力側でも意味はないが ASCII に含まれるものを検出したことが，暗号処理以外のブロックを検出した原因と考えられる．

そこで，他のエンコードを適用することで，出力側で印刷可能文字と認識される確率が下がると考えられる．そのため，今後は，バッファのエンコードを確認し，適切に対処する必要があると考える．

5.2 ブロックの範囲について

今回，ブロックの範囲を制限せずに実行トレースから抽出した．その結果，範囲が極端に広いブロックと，狭いブロックが抽出された．ブロックの定義として今回関数を用いたが，単純に call-ret で定義すると無制限に広くなる．ステップ数や，アドレスの範囲があまりに広いものは除外することが有効と考えられる．

また，ブロックを構成する実行トレース行の数も暗号処理の判別に用いることができると考える．暗号処理は通常繰り返し処理を行うなど，処理がある程度長くなる傾向がある．そのため，表 1 の No1 のように，11 行分しか実行トレースを取得していないブロックに関しては暗号処理である可能性が低く，候補から除外できる．

5.3 送信内容が文字列ではない場合

攻撃者が送信する情報は，文字列とは限らず，画像や word ファイルといったファイルを送信することも考えられる．そこで，それらのファイルごとの特徴を知っておき，比較することで暗号化関数の判定に役立つと考えられる．

6. むすび

本稿では，マルウェアの実行トレースから，暗号処理を行っている部分を特定する手法について，実験と考察を行った．今回，本手法によって，暗号処理を行う範囲を絞り込むことに成功した．今後は，誤検知と閾値に関して検討し，暗号処理候補の抽出精度を向上させる．また，送信内容が文字列でない場合に対しても検討し，課題を解消していく．

表 4　TEA で検出されたブロックの解析結果

No1	“__SEH_epilog4”関数である．これは例外処理のハンドラを扱う関数であり，暗号処理とは無関係である．
No2, 4	このブロックが示す関数が，引数として”Input text”や，”Key”を含んでいたため，文字の出力関数と考えられる．
No3	解析した結果，TEA の暗号アルゴリズムを実装した部分である．
No5	13210a0 はエントリポイントであるため，このブロックは main 関数である．

表 5　RC4 で検出されたブロックの解析結果

No1, 2, 3, 6	このブロックが示す関数の，main 関数での呼び出され方を調べると，標準出力への文字出力関数であると考えられる．
No4	RC4 で暗号処理を行うために必要な，乱数生成処理を行っている．
No5	解析した結果，RC4 の暗号処理を実装した部分である．
No7	文字の扱いに関する処理を行っていると考えられる．暗号処理とは無関係である．
No8	アドレス f81420 はエントリポイントであるため，このブロックは main 関数である．

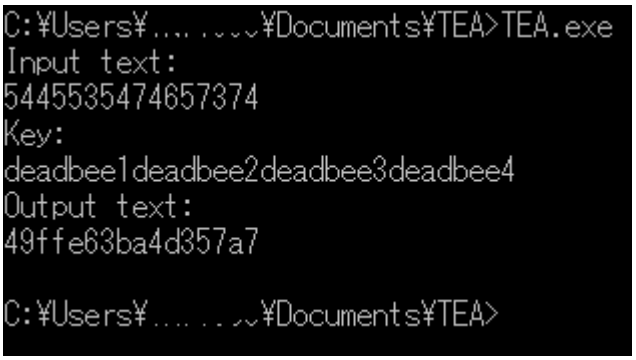


図 4　TEA の実行ファイルを実行した様子

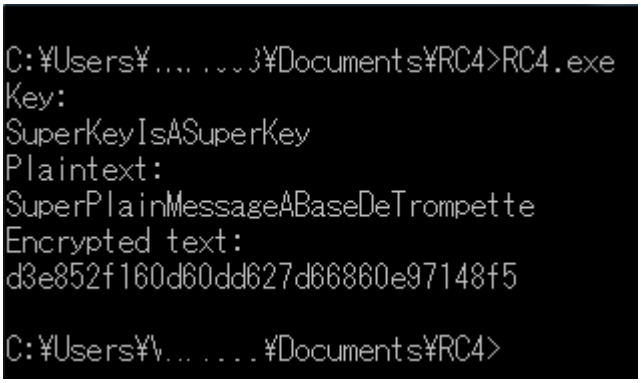


図 5　RC4 の実行ファイルを実行した様子

ⓒ 2016 Information Processing Society of Japan

Vol.2016-DPS-166 No.24
Vol.2016-CSEC-72 No.24
2016/3/4

参考文献

1) 警察庁: 平成27年上半期のサイバー空間をめぐる脅威の情勢について,
http://www.npa.go.jp/kanbou/cybersecurity/H27_kami_jousei.pdf
2) Joan Calvet, Jose M. Fernandez and Jean-Yves Marion: Aligot: Cryptographic Function Identification in Obfuscated Binary Programs, Proceedings of the ACM Conference on Computer and Communications Security (2012).
3) Felix Grobert, Carsten Willems and Thorsten Holz: Automated Identification of Cryptographic Primitives in Binary Programs, Proceedings of the 14th International Conference on Recent Advances in Intrusion Detection (2011).
4) Wang, Z., Jiang, X., Cui, W., Wang, X., Grace, M.: ReFormat: Automatic ReverseEngineering of Encrypted Messages, Proceedings of the European Symposium on Research in Computer Security (2009).
5) 山本匠, 西川弘毅, 河内清人, 中嶋純子, 桜井鐘治: 暗号ロジック特定手法の提案, コンピュータセキュリティシンポジウム 2014, pp. 835- 842 (2014).
6) 山本匠, 西川弘毅, 河内清人, 中嶋純子, 桜井鐘治: 暗号ロジック特定手法の提案 その２, 2015 暗号と情報セキュリティシンポジウム, (2015).
7) Intel: Pin - A Dynamic Binary Instrumentation Tool,
http://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool
8) Aligot Project: aligot - Cryptographic function identification in obfuscated programs, r120,
http://code.google.com/p/aligot/source/browse/trunk/vanilla/testBinaries/TEA/TEA.exe
9) Aligot Project: aligot - Cryptographic function identification in obfuscated programs, r120,
https://code.google.com/p/aligot/source/browse/trunk/vanilla/testBinaries/RC4/RC4.exe

ⓒ 2016 Information Processing Society of Japan