暗号ロジック特定手法の提案

山本 匠　西川 弘毅　河内 清人　中嶋 純子　桜井 鐘治

三菱電機株式会社 情報技術総合研究所
〒247-8501 神奈川県鎌倉市大船 5-1-1
{ Yamamoto.Takumi@ak | Nishikawa.Hiroki@dn | Kawauchi.Kiyoto@ab
| Nakajima.Junko@dc | Sakurai.Shoji@ab }.MitsubishiElectric.co.jp

あらまし 昨今のマルウェアは暗号技術を活用し通信を秘匿する. そのため企業内でマルウェア感染が発生した際に, 攻撃者がどのような情報を窃取したのかを追跡することが困難となっている. これに対応するためには, マルウェアが通信の秘匿に利用した暗号ロジックを見つけ, 暗号化された通信を復号する必要がある. この作業には通常, マルウェアのバイナリを解析する必要があり手間暇を要する. そこでマルウェアを実行した際の実行トレースから, 暗号ロジックによく見られる特徴やマルウェアが暗号ロジックを利用する際の特徴を探すことで, 暗号ロジックを自動的に特定する手法を提案する.

Proposal for Cryptographic Function Identification

Takumi Yamamoto　Hiroki Nishikawa　Kiyoto Kawauchi
Junko Nakajima　Shoji Sakurai

Mitsubishi Electric Corporation Information Technology R&D Center.
5-1-1 Ofuna, Kamakura, Kanagawa 247-8501, Japan
{ Yamamoto.Takumi@ak | Nishikawa.Hiroki@dn | Kawauchi.Kiyoto@ab
| Nakajima.Junko@dc | Sakurai.Shoji@ab }.MitsubishiElectric.co.jp

Abstract Recent malwares tend to use encryption on communication to avoid being exposed. Once an infected PC starts to communicate with unknown hosts via the Internet, its organization has to investigate which information has been leaked by the malware. Usually such an investigation involves malware binary analysis, which is vastly complicated and time-consuming as well, to figure out cryptographic functions and keys used in the fraudulent communication. To facilitate the investigative task, we propose a method to automatically detect cryptographic functions in the execution traces output from malwares capturing characteristics precisely found commonly in cryptographic implementations.

1　はじめに

近年新しいセキュリティ脅威として，特定の組織をねらい，執拗に攻撃を行う Advanced Persistent Threat（APT）と呼ばれる“標的型攻撃”が顕在化している[1,2]. APT では，メールによって標的とする組織の端末にマルウェアを感染させ，感染したマルウェアが外部の攻撃者のサーバと通信を行い，新しい攻撃プログラムのダウンロードや組織システム内の機密情報の送信を行う[3,4].

このようなセキュリティインシデントが発生した場合，組織ではインシデントの原因や被害の調査，対応策の検討，サービスの復旧，再発防止策の実施などのインシデントレスポンスが行われる. 顧客やビジネスパートナーによっては，マルウェア感染によって何が漏えいし，あるいは何が漏えいしなかったのかを，明確にする必要がある.

インシデントの原因や被害の調査を行うにあたり，パソコン，サーバ，ネットワーク機器などが生成したログやネットワーク上で記録されたパケットを解析し，マルウェアの侵入経路，感染端末，アクセスした情報，攻撃者からの命令，外部に送信した情報などを調査するネットワークフォレンジックが重要な役割を担う[5-9].

ところが，昨今のマルウェアは暗号技術を活用し通信を秘匿する. そのためネットワークフォレンジックを行ったとしても，攻撃者から送られる命令が何であり，どのような情報が外部に送信されたのかを追跡することが困難となっている[10].

したがって，マルウェアが通信の秘匿に利用した暗号ロジックと鍵を見つけ，暗号化された通信を復号する必要がある. 通常この作業にはマルウェアのバイナリを解析する必要があり，膨大な手間と時間を要する.

この問題に対応するために，マルウェアを実行して得られる実行トレースを解析して，マルウェアが利用している暗号ロジックを特定する技術が提案されている[11,12]. しかし既存研究には，解析に時間を要する，大量に暗号ロジックの候補が出力される（誤検知），未知の暗号ロジックを特定できない(検知漏）などの課題がある.

本稿では，暗号ロジックによく見られる特徴や，マルウェアが暗号ロジックを利用する際の特徴が，実行トレースに含まれるかを確認することで，誤検知や未知の暗号ロジックの検知漏れを軽減する手法を提案する.

以下，2 章でマルウェアの暗号ロジックの特定に関する既存研究を紹介し，3 章で提案手法のコンセプトを説明する. 4 章で提案手法について考察し，5 章で本稿をまとめる.

2　既存研究

著者がこれまでに調査した限りでは，暗号ロジックと鍵の特定に関する既存技術は，シグネチャを用いるアプローチ[13-16]と実行トレースを用いるアプローチ[11,12]の 2 種類に大別される. 以下では，両解析技術について概説する.

2.1　シグネチャを用いるアプローチ

シグネチャを用いるアプローチでは，あらかじめ暗号ロジック特有のコードやデータのパターンを DB に登録しておき，ハードディスク（HDD）上のプログラムのイメージにパターンが含まれているかを確認する. このパターンのことを一般的にシグネチャと呼ぶ. シグネチャは暗号アルゴリズムごとに用意される.

シグネチャを用いるアプローチは，静的解析を用いる手法に分類される. 静的解析とは，プログラムを実行せずに，HDD 上のプログラムのイメージだけから，プログラムの特徴を抽出し，解析を行う手法のことである. シグネチャを用いて既知の暗号ロジックを特定する既存技術としては，draca[13]，SnD Crypto Scanner[14]，Hash & Crypto Detector[15]，PEiD[16]が知られている.

同アプローチのメリットは，高速な解析が可能なこと及びバイナリに関する高度な知識や技

術をユーザに必要としない点である．一方，デメリットは，実行ファイルに圧縮や暗号化などの難読化が適用されていると，正しく解析することができない点である．また同アプローチは，未知の暗号ロジックの検知はできない．

2.2　実行トレースを用いるアプローチ

実行トレースを用いるアプローチとは，プログラムを実際に動かし，実行された命令，実行時のレジスタやメモリの値などの情報を逐次記録し，記録された情報を解析する手法である．プログラムを実際に動かすため，動的解析の 1 つと言える．

動的解析とは，プログラムを実際に実行し，実行された命令，実行時のレジスタやメモリの値などの情報を活用し，プログラムの特徴を解析する手法である．実行トレースを用いて実行ファイルの中の暗号ロジックを特定する既存研究としては，文献[11,12]が知られている．

これらのツールは Intel の Pin[17]を使って収集した実行トレースを解析する．実行トレースには，実行された命令，実行時のレジスタやメモリの値などの情報が記録される．実行トレースを解析することで暗号ロジックを特定する．実行トレースを収集する手法としては，Valgrind[18]や QEMU[19]を使う方法も知られている．

同アプローチのメリットは，メモリに展開された情報から平文や鍵などの暗号ロジックに入力された引数も抽出できる点である．また，同アプローチでは，圧縮や暗号化などの難読化が適用された実行ファイルも解析することができる．

同アプローチのデメリットは，実行トレースが膨大になるため，解析に時間を要する点である．

近年のマルウェアの多くは，圧縮や暗号化などの難読化が適用されているものがほとんどである．そのため本研究では，実行トレースを用いるアプローチを採用する．以下では，実行トレースを用いる方式の代表的な既存研究として，kerckhoffr [11]及び Aligot[12]を紹介する．

● kerckhoffr

本手法では，プログラム内のループ構造を探し，ループを含むブロックに対して，暗号ロジック特有の命令列や定数の組みをパターンとしたパターンマッチング，入力と出力のエントロピの差が大きい処理の抽出，を行うことで暗号ロジックの特定を行う．暗号ロジック特有の命令列とは，算術演算やビット演算の並びをいう．

本手法は非常に多くの暗号ロジックの候補を出力するため，さらに手作業で候補を絞り込む必要がある．

● Aligot

本手法では命令列や定数のパターンには注目しない．本手法では，プログラム内のループ構造を探し，ループを含むブロックに対して，入出力を抽出する．抽出した入力を既知の暗号ロジックの実装に入力し，同暗号ロジックを実行する．同暗号ロジックから得られた出力が，同ブロックの出力と等しければ，同ブロックは同暗号ロジックであると断定する．

本手法は誤検知を起こさないが，解析にメモリと時間を大量に消費する．また未知の暗号ロジックは見つけることができない．

3　提案方式

3.1　提案方式の概要

図 1に提案方式の流れを記載する．提案方式では，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．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．

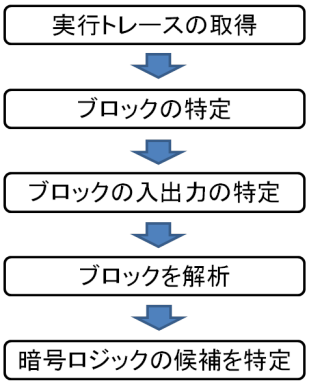

図 1 提案方式の流れ

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．

暗号ロジックによく見られる特徴や，マルウェアが暗号ロジック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，暗号ロジックの候補を特定する．

なお提案方式は，検査対象プログラムが必ず暗号ロジックを実行することを前提としている．次節から，図 1の各ステップについて詳しく説明していく．

3.2 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 Dynamic Instrumentation Tool である Pin[17]を利用する．Pin では，1 命令実行単位で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．

実行トレースとして最低限必要な情報を以下に記す．

- 実行した命令のアドレス，オペコード，オペランド
- 操作されたレジスタ，その値，操作情報（READ/WRITE）
- 操作されたメモリのアドレス，その値，操作情報

3.3 ブロックの特定

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．暗号ロジックは，ブロックから作られることが一般的である．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．

● ループ

ループの特定には文献[12,20]で紹介されている方法を利用する．文献[20]では，実行トレース上の後方ジャンプ（Backward Jump）を探すことでループを検出する．後方ジャンプ先から後方ジャンプ元までの命令列をループのボディとする．

文献[12]では，実行トレース上の同じ命令列の繰返しを探すことでループを検出する．繰り返されている命令列をループのボディとする．

● 関数の特定

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関数のボディとする．

3.4 ブロックの入出力の特定

暗号ロジックは，入力として平文と鍵を受け取り，暗号文を出力する．本研究では暗号ロジック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特定する[12]．

● 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

・ ブロックの中で，自身と隣接するメモリも同じ命令で参照されるメモリ

● 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）
・ ブロックの中で，自身と隣接するメモリも同じ命令で定義されるメモリ

3.5 ブロックの解析

提案方式では，以下に示す方法により，ブロックを解析する．以下の方法では，暗号ロジックの入出力の特徴やマルウェアの暗号ロジックの利用方法に着目する．

なお解析対象のブロックを絞り込むために，2.2 節で挙げた暗号ロジック特有の命令列や定数の組みをパターンとしたパターンマッチングを事前に適用することも可能である．

●暗号ロジックの入出力特徴に基づく解析

暗号ロジック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，暗号ロジックの出力に，印刷可能な文字列が含まれることは多くはないと考えられる．

印刷可能文字列とは，改行復帰文字または空文字（ヌル文字）で終わる C 個またはそれ以上の印刷可能な文字の連続のことである．C は調整可能なパラメータである．

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，暗号ロジックの入力，すなわち，平文には，印刷可能文字列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列の割合を比較することで，同ブロックが暗号ロジックの候補かどうかを判定する（図 2）．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．

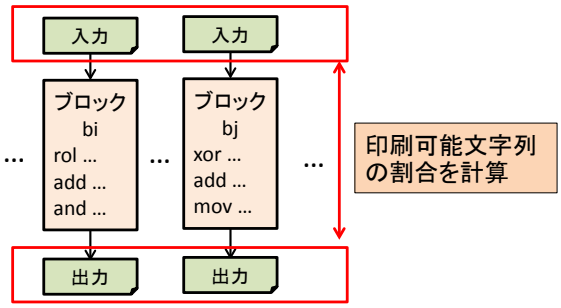

図 2 解析イメージ 1

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 I_b をブロック b の入力の集合，O_b をブロック b の出力の集合とし，I_b と O_b から印刷可能文字列の割合を計算し，それぞれ Ri，Ro とする．
④ |Ri-Ro|＞θ ならば，b を暗号ロジック（または復号ロジック）とする．
⑤ ②を実行する．

ステップ③について補足で説明する．印刷可能文字列の割合は，入力（または出力）から得られる印刷可能文字列の長さの総和を入力（または出力）の長さで割ったものである．

●マルウェアの暗号ロジック利用方法に基づく解析

マルウェアは暗号化したデータを Base64 エンコーダやパーセントエンコーダなどのエンコーダで印刷可能文字列に変換し，HTTP 通信のボディやヘッダに同文字列を埋め込むことがある．すなわち出力がエンコードされるブロックは，暗号ロジックの可能性が高いと考えられる．そこで，各ブロックの出力を既知のアルゴリズムでデコードし，デコードに成功したデータを出力に持つブロックを，暗号ロジックの候補と判定する（図 3）．解析手順は次のとおりである．

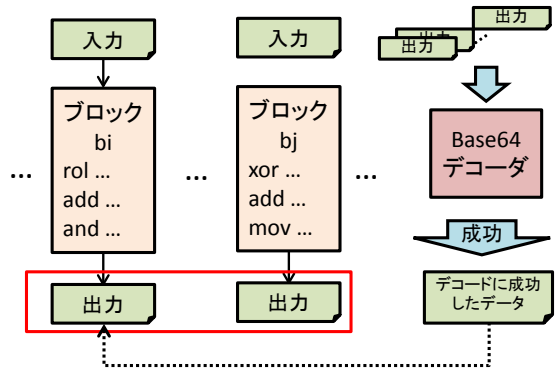

図 3 解析イメージ 2

① 確認済みブロック集合 UB を空にする.
② ブロック b_1($b_1 \in B$, $b_1 \notin$ UB)を選択する. UB に b_1 を追加し, 確認済み出力集合 UO を空にする. 該当するブロックが無ければ終了する.
③ O_{b1} をブロック b_1 の出力の集合とし, o ($o \in O_{b1}$, $o \notin$ UO)を選択する. UO に o を追加する. 該当する出力が無ければ②を実行する.
④ o を既知のデコードアルゴリズムでデコードする. デコードに成功すれば⑤, しければ③を実行する.
⑤ o のデコード結果と同じ値の出力を持つブロック b_2($b_2 \in B$, $b_2 \neq b_1$)があれば, ブロック b_2 を暗号ロジックの候補とし②を実行する. 該当するブロックが無ければ③を実行する.

4 考察

暗号ロジックの検知性能とオーバーヘッドの観点から提案方式について議論する.

4.1 検知性能

● ブロックの検出について

提案方式は, 関数やループなどのブロックを暗号ロジックの候補とし, さらに解析を進める. そのためブロックの検出精度が悪ければ, 暗号ロジックを検知できない可能性がある. 難読化により非常に複雑な制御フローを持ったマルウェアが現れる可能性もある.

現状では, そのようなマルウェアから正確にブロックを検出することは困難と考えられる. そのため, 今後はブロックによらない方法も検討していく必要がある.

● 印刷可能文字列の割合について

PDF や WORD などの文書ファイルでは, 文書内のテキストが圧縮されていることがある. そのような文書ファイルは, 印刷可能文字列の割合が低いと考えられ, 暗号ロジックの入出力特徴に基づく解析手法が機能しないことが予想される. これについては, 入力される情報からファイルタイプを判定し, ファイルタイプに応じたデコードを行った上で, 印刷可能文字列の割合を計算するなどの工夫が必要である.

●マルウェア独自のエンコーダについて

マルウェアが独自のエンコーダを用いた場合, マルウェアの暗号ロジック利用方法に基づく解析手法は機能しない. 例えばマルウェアは Base64 エンコーダの変換テーブルをカスタマイズするだけで, 新しいエンコーダを簡単に作成することができる. 全てのエンコーダを用意することは不可能ではあるが, マルウェアがよく利用するエンコーダについて定期的に調査し, 適切なデコーダを用意しておく必要がある.

4.2 オーバーヘッド

実行トレースが多くなればなるほど, 解析しなければならない命令や検出される基本ブロックの数は増大する. すなわち, 実行トレースの増大は解析に必要な時間やメモリの増大につながる. そのため, ファイル操作処理や通信処理など暗号化や復号と関係が深そうな実行トレースのみ解析するなどの効率化するための工夫も必要である

4.3 マルウェアの制御

提案方式は, 検査対象プログラムが必ず暗号ロジックを実行することを前提としている. そ

のため，耐解析機能を持つマルウェアに対しては，耐解析機能を無効にする作業が必要となる．また攻撃者から期待するコマンドを受け取らないと暗号ロジックを実行しないマルウェアも存在する可能性もある．これは実行トレースを用いるアプローチにおける共通の課題である．

これに対しては，逆アセンブラやデバッガを駆使し，暗号ロジックを実行するようにマルウェアにパッチを当てることで対応することができる．ただし，このような作業はアセンブリ言語や OS に関する専門的で高度な知識や技術を必要とするだけでなく，解析に手間と時間を要する．

そのため，文献[22]のように，マルウェアを安全かつより現実に近い環境で実行させ，マルウェア本来の動作をさせることが重要になる．

5 まとめと今後の課題

本稿では，マルウェアの実行トレースから暗号ロジックを特定する手法のコンセプトを提案した．暗号ロジックの本質的な特徴やマルウェアの暗号ロジックの利用の仕方に着目することで，暗号ロジックの誤検知を抑えることが期待できる．提案手法は，既知の暗号ロジックの実装やめ暗号ロジック特有のコードやデータのパターンを基にしない．さらに提案手法では，暗号ロジックの本質的な特徴，すなわち，入出力の関係に着目することで，未知の暗号ロジックでも検知可能であると考えられる．

今後は提案方式のプロトタイプを実装し，評価実験を通じ，提案方式の実現可能性を確認する予定である．

参考文献

[1] シマンテック, "「標的型攻撃」に備えるーサイバー攻撃: 標的型攻撃とは，APT とは", http://www.symantec.com/ja/jp/theme.jsp?themeid=apt_insight（2014 年 7 月 23 日確認）.

[2] kaspersky , "Advanced Persistent Threat（APT）攻撃：今までにない高度なマルウェア", http://www.kaspersky.co.jp/downloads/pdf/advanced-persistent-threats-not-your-average-malware_kaspersky-endpoint-control-white-paper_jp.pdf （ 2014 年 7 月 23 日確認）.

[3] McAfee, "Know Your Digital Enemy Anatomy of a Gh0st RAT", http://www.mcafee.com/sg/resources/white-papers/foundstone/wp-know-your-digital-enemy.pdf.（2014 年 7 月 23 日確認）.

[4] Securelist, "The Icefog APT: A Tale of Cloak and Three Daggers", http://www.securelist.com/en/blog/208214064/The_Icefog_APT_A_Tale_of_Cloak_and_Three_Daggers（2014 年 7 月 23 日確認）.

[5] Karen Kent, Suzanne Chevalier, Tim Grance and Hung Dang, "Guide to Integrating Forensic Techniques into Incident Response", NIST, Special Publication 800-86.

[6] Paul Cichonski, Tom Milar, Tim Grance and Karen Scarfone, "Computer Security Incident Handling Guide", NIST, Special Publication 800-61 Revision 2.

[7] Sherri Davidoff and Jonathan Ham, "Network Forensics: Tracking Hackers through Cyberspace", PRENTICE HALL.

[8] Joe Fichera and Steven Bolt, "Network Intrusion Analysis: Methodologies, Tools, and Techniques for Incident Analysis and Response", FICHERA BOLT.

[9] Steven Anson, Steve Bunting, Ryan Johnson and Scott Pearson, "Mastering Windows Network Forensics and Investigation", SYBEX.

[10] Shawn Denbow, Matasano Security, "pest control: taming the rats", http://www.matasano.com/research/PEST-CONTROL.pdf（2014 年 7 月 23 日確認）.
[11] Felix Gröbert, 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.
[12] Joan Calvet, Jose M. Fernandez and Jean-Yves Marion, "Aligot: Cryptographic Function Identification in Obfuscated Binary Programs", Proceedings of the 19th ACM Conference on Computer and Communications Security, 2012.
[13] Literatecode, "draft crypto analyzer", http://www.literatecode.com/draca（2014 年 7 月 23 日確認）.
[14] Collaborative RCE Tool Library, "SnD Crypto Scanner (Olly/Immunity Plugin)", http://www.woodmann.com/collaborative/tools/index.php/SnD_Crypto_Scanner_(Olly/Immunity_Plugin)（2014 年 7 月 23 日確認）.
[15] Collaborative RCE Tool Library, Hash & Crypto Detector, http://www.woodmann.com/collaborative/tools/index.php/Hash_%26_Crypto_Detector（2014 年 7 月 23 日確認）.
[16] peid.info, "Peid 0.95", http://www.peid.info/（2014 年 7 月 23 日確認）.
[17] Intel, "Pin - A Dynamic Binary Instrumentation Tool", http://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool.（2014 年 7 月 23 日確認）.
[18] Valgrind Home, "Vargrind", http://valgrind.org/（2014 年 7 月 23 日確認）.
[19] QEMU open source processor emulator, "Main Page", http://wiki.qemu.org/Main_Page（2014 年 7 月 23 日確認）.
[20] Jordi Tubella and Antonio Gonzalez, "Control Speculation in Multithreaded Processors through Dynamic Loop Detection", Proceedings of the Fourth International Symposium on High-Performance Computer Architecture , 1998.
[21] Securelist, "GpCode-like Ransomware Is Back", https://www.securelist.com/en/blog/333/GpCode_like_Ransomware_Is_Back（2014 年 7 月 23 日確認）.
[22] 瀬川達也，神薗雅紀，星澤裕二，吉岡克成，松本勉，"Man-in-the-Browser 攻撃を行うマルウェアの安全な動的解析手法,"研究報告コンピュータセキュリティ(CSEC)，2013-CSEC-61, no. 8, pp. 1- 8, 2013.