Computer Security Symposium 2016
11 - 13 October 2016

マルウェアが行う暗号処理の鍵生成源特定手法の提案

西川 弘毅[†1]　祢宜 知孝[†1]　河内 清人[†1]

概要：昨今のマルウェアは，活動内容を隠すために，暗号技術，特に共通鍵暗号を用いて通信を秘匿する．暗号化された通信ログでは，マルウェアが窃取した情報の特定が困難である．そこで，通信ログの復号のためにマルウェアが利用している暗号鍵を特定する必要がある．しかしマルウェアは，暗号通信で用いる鍵を，環境毎に異なるものを利用する場合がある．このようなマルウェアでは，被害環境と解析環境とで生成される鍵が異なるため，容易には鍵が特定できず，行われた通信を復号することができない．そこで，マルウェアが環境毎に生成する鍵の元となる，鍵生成源を特定する手法を提案する．

キーワード：マルウェア，暗号処理，フォレンジック

Proposal of identification to key generator using in encryption malware use

Hiroki Nishikawa[†1]　Tomonori Negi[†1]　Kiyoto Kawauchi[†1]

Abstract: Recently malwares conceal they communication using common key encryption. We find difficult to identifying which information malware theft from encrypted communication log. Therefore, for decrypt encrypted communication log, we need to identify encryption key used by malware. However, some malwares use different encryption key in each environment. Because those malwares generate different encryption key in affected and analysis environment, decrypting malware communication is difficult problem. We propose identification method to key generator using in encryption malware use.

Keywords: Malware, Encryption, Forensics

1. はじめに

近年，機密情報の窃取を目的とする，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．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．この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，メール受信者が端末において同文書を開いた瞬間，その端末はマルウェアに感染してしまう．攻撃者はこのマルウェアをインターネット上の 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し，C&C サーバへアップロードすることで目的を達成する．このような機密情報の漏えい被害の深刻化を背景に，マルウェアに感染したパソコンやサーバ等が生成したログを解析することで，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，通信データを共通鍵暗号により暗号化することで，通信データを秘匿するものがある．このようなマルウェアの通信データは，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，マルウェア解析者は，マルウェアが通信データの暗号化で用いている暗号アルゴリズムと，暗号化に用いる暗号鍵とを特定して，暗号化された通信を復号する作業が必要となる．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るため，一般に膨大な手間と時間が必要となる．そこで，自動的にマルウェアの暗号アルゴリズムの特定を行う手法[1]や，暗号鍵を特定する手法[2]が研究されている．

従来の技術では，マルウェアが利用している暗号アルゴリズムが特定できても，動的に鍵を生成するマルウェアにおいては，復号対象の通信ログに対応する鍵を特定する事ができないという課題があった．ここで動的な鍵生成とは，暗号に利用する鍵がマルウェアにハードコードされずに，マルウェアが活動する環境内の情報等を元に作成し，利用することと定義する．

動的に鍵を生成するマルウェアは，例えば，感染端末上の IP アドレスを，暗号鍵を生成するシードとして用いて，暗号に利用する鍵を生成し，盗み出す機密ファイルの暗号化を行う．この場合，異なる端末では異なる鍵が生成され，暗号に利用される．そのため，被害が発生した端末の鍵（以降，被害鍵と呼ぶ）と，マルウェア解析環境での鍵（以降，解析鍵と呼ぶ）とが異なる．ここで，漏えい情報は被害環境で発生しているため，被害鍵により暗号化されている．そのため，解析環境で手に入る解析鍵では，暗号化された通信ログを復号することができない．

†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.

```
401055!mov esi, ecx!RR_ecx_63c02188!WR_esi_63c02188
401057!shr esi, 0x5!RR_esi_63c02188!WR_esi_31e010c
40105a!add esi, dword ptr [ebp-0x8]!RM_12ff14_4_deadbee4!RR_ebp_12ff1c_esi_31e010c!WR_esi_e1cbbff0
40105d!mov edi, ecx!RR_ecx_63c02188!WR_edi_63c02188
40105f!shl edi, 0x4!RR_edi_63c02188!WR_edi_3c021880
401062!add edi, dword ptr [ebp-0xc]!RM_12ff10_4_deadbee3!RR_ebp_12ff1c_edi_3c021880!WR_edi_1aafd763
401065!xor esi, edi!RR_esi_e1cbbff0_edi_1aafd763!WR_esi_fb646893
401067!lea edi, ptr [edx+ecx*1]!RR_ecx_63c02188_edx_28b7bd67!WR_edi_8c77deef
40106a!xor esi, edi!RR_esi_fb646893_edi_8c77deef!WR_esi_7713b67c
40106c!sub eax, esi!RR_eax_a3651d14_esi_7713b67c!WR_eax_2c516698
40106e!mov esi, eax!RR_eax_2c516698!WR_esi_2c516698
```

図 1 Aligot での実行トレースの例

Figure 1 Example of execution trace used by Aligot.

命令のアドレス	命令 (オペコード)	命令対象 (オペランド)	メモリ・レジスタ アクセス情報

図 2 実行トレース一行分の構成図

Figure 2 Structure of record in execution trace.

このように，従来の技術では，解析鍵の特定は可能であったが，被害鍵の特定はできないという課題があった．

そこで本稿では，被害鍵を特定するために，被害鍵を生成するために必要な情報である鍵生成源の特定手法を提案する．

2. 関連研究

マルウェアの検体から，マルウェアの暗号アルゴリズムを特定する手法[1]や，マルウェアが利用している暗号鍵を特定する手法[2]が研究されている．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，マルウェアの暗号処理を構成する暗号ブロックを抽出し，解析することでマルウェアが利用している暗号アルゴリズムを特定する．Aligot での実行トレースの例を図 1 に示す．また，実行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．この構成図を図 2 に示す．一般に，実行トレースを利用した暗号アルゴリズムの特定は，リバースエンジニアリングによる静的解析と比較して難読化されたマルウェアに対しても解析が容易であるという特徴があるため，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法といえる．実行トレースの抽出には Pin[3]等のツールが用いられる．

以下に，マルウェアが利用する暗号アルゴリズムの特定手法[1]と，マルウェアが利用する暗号鍵の特定手法[2]について説明する．

文献[1]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，暗号アルゴリズムの特定では，マルウェアが用いる暗号アルゴリズムが既知の暗号アルゴリズムであることが多いという仮定で，暗号ブロックの入出力と一致するような既知暗号アルゴリズムの入出力を探し出し，一致した既知暗号アルゴリズムを暗号アルゴリズムとして判定している．

文献[2]の手法では，マルウェアに復号処理を行わせ，その処理中にアクセスされたマルウェア内部のメモリ間の依存関係を解析することで鍵候補の抽出を行う．

この手法で特定する鍵は，C&C サーバからマルウェアに対する制御メッセージの復号鍵である．従って，この手法では，C&C サーバから送られる暗号化されたメッセージに対する復号鍵が，標的組織内から制御サーバへ情報を送信する際の暗号鍵にも利用されるという仮定に基づいた手法である．

しかし，これらの手法では，動的に鍵を生成するマルウェアに対して，被害環境での鍵である被害鍵を特定する事ができず，マルウェアの暗号通信を復号できないという課題があった．

3. 動的な鍵生成を行うマルウェア

動的な鍵生成について図 1 から図 3 を用いて説明する．

図 1 は，マルウェアが動的に鍵を生成する例を示す図である．

この例で示すマルウェアは，感染端末上の IP アドレスを，暗号鍵を生成するシードとして用いて，暗号に利用する鍵を生成し，盗み出す機密ファイルの暗号化を行う．この場合，異なる端末では異なる鍵が生成され，暗号処理に利用される．

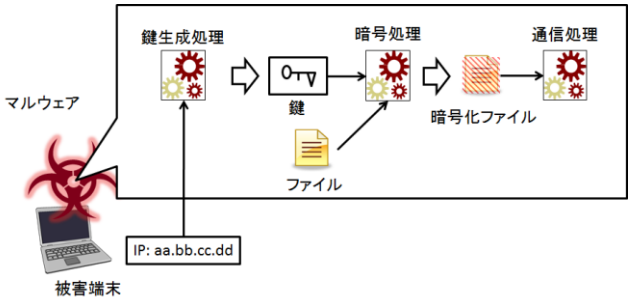

図 3　マルウェアが動的に鍵を生成

Figure 3　Malware generate key dynamically.

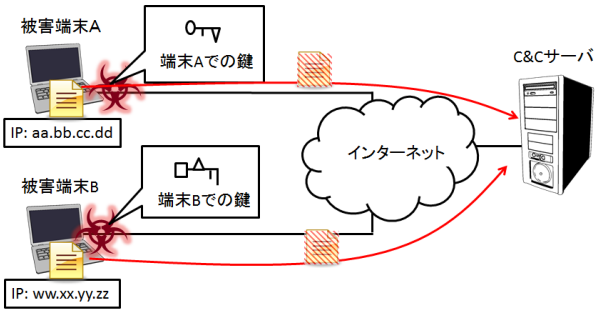

図 4　環境毎に異なる鍵が生成

Figure 4　Different keys are generated in each environment.

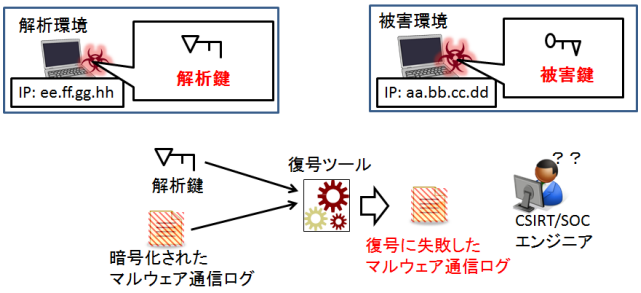

図 5　解析環境と被害環境が異なるために復号に失敗

Figure 5　Decryption is failed due to difference in affected and analysis environment.

図 2 は，異なる鍵がそれぞれの被害端末で生成される様子を示す図である．被害端末 A と被害端末 B とは，同一のマルウェアに感染しているが，マルウェアの暗号処理で用いられる鍵は異なる．

図 3 は，マルウェアにより暗号化された暗号化ファイルの復号を依頼された SOC(Security Operation Center)/CSIRT(Computer Security Incident Response Team)エンジニアが，解析鍵では復号できない様子を示している．図 3 に示すように，マルウェアを解析する場合，被害が発生した被害環境における被害鍵と，マルウェアの解析環境での解析鍵とは異なる．漏えい情報は被害環境で発生しているため，被害鍵により暗号化されている．そのため，解析環境で手に入る解析鍵では，暗号化された通信ログといった機密ファイルを復号することができない．マルウェア

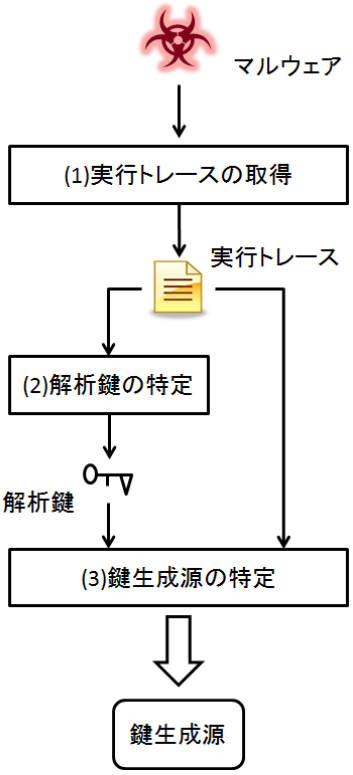

図 6　マルウェアの鍵生成源特定の流れ

Figure 6　Flow of malware's key generator.

により暗号化された暗号化ファイルを復号するために，マルウェアが利用する暗号アルゴリズムと暗号鍵とを特定する必要がある．しかし，マルウェアが感染した端末の環境情報を利用して鍵を生成する場合，解析環境で得られる鍵では被害環境で発生した暗号化ファイルを復号することができない．そこで，本稿では，解析環境において特定できた鍵の情報を元に，どの環境情報を鍵生成源として利用しているかを特定し，暗号通信の復号にかかる手間を削減する手法を提案する．

4. 提案手法

4.1 全体概要

本手法を用いて，マルウェア検体から鍵生成源を特定する流れを図 6 に示す．また，本手法の各処理について説明する．

(1)　実行トレースの取得

マルウェアを解析環境で実行し，暗号処理を行った際の実行トレースを，Pin 等のツールにより取得する．実行トレースを取る際には，call した際の関数名を記録するように注意する．

(2)　解析鍵の取得

実行トレースを解析し，マルウェアの解析鍵を特定する．本処理は，例えば 2 章で示した文献[2]の手法により実現す

ることができる．

(3)　鍵生成源の特定

前処理で特定した解析鍵と，実行トレースを元に，マルウェアの鍵生成源を特定する．本処理については次節で詳細に説明する．

以上の処理によりマルウェアから鍵生成源を特定する事が可能となる．ここで，本手法の目的は鍵生成源の特定である．そのため，マルウェアから実行トレースを取得する処理から鍵生成源を特定する処理までを自動化することが理想的ではあるが，実行トレースの取得と，解析鍵の取得に関しては，手動による解析を一部行っても良いとする．

4.2 鍵生成源の特定

本節では，鍵生成源の特定手順について説明する．ここで，実行トレースと，実行トレースを解析することによって得られる解析鍵の二つは既に取得されているものとする．

鍵生成源特定の流れを以下に示す．

(1) 実行トレースにおける解析鍵の位置を取得する．
(2) 解析鍵と依存関係にある命令を追跡し，終端までのアセンブルリストを得る．
(3) 得られたアセンブルリストの call 命令が，外部情報を参照するものかを判定する．
(4) アセンブルリストが外部情報源を参照するものである場合，そのアセンブルリストを鍵生成源として特定する．

続いて，それぞれのステップの詳細を説明する．

(1)　解析鍵特定部が特定した解析鍵の情報として，解析鍵が実行トレース上でどこに存在しているかの情報を受け取る．図 7 は，解析鍵特定部からの情報を元に，実行トレース上でどのメモリが解析鍵であるかを特定している様子を示している．ここでは，解析鍵が 16 進数で「AAAAA」であり mem2 に格納されていた場合を考える．ここで，mem1，mem2 とはメモリのある領域を指している．

(2)　特定した解析鍵の位置 mem2 から，解析鍵と依存関係のある命令を，テイント解析手法を用いて追跡していく．図 8 に，解析鍵と依存関係のある情報をテイント解析により求める様子を示す．

まず，mem2 は ecx の値が格納されているため，ecx の値に依存している．次に，その前段で ecx は eax の値と加算した結果を格納している．更に，その前段で eax は mem1 の値を格納している．このように依存関係を辿っていくと，最終的に mem2 の値は mem1 の値に依存していることが分かる．このようなテイント解析を，実行トレース全体に渡って行った結果，図 9 のようなアセンブルリストを得る．

(3)　このアセンブルリストにおいて，call 命令の行を抽出し，call 命令が呼び出している関数と同一のものが外部情報参照関数データベースにあるか問い合わせる．

(4)　問い合わせた関数が外部情報参照関数データベースに含まれていた場合，問い合わせた関数を呼び出しているアセンブルリストを鍵生成源とする．

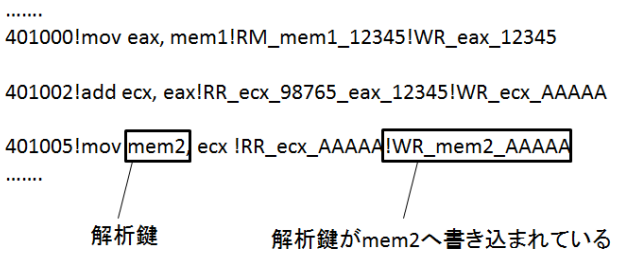

図 7　実行トレース上での解析鍵の位置

Figure 7　Location of analysis key on execution trace.

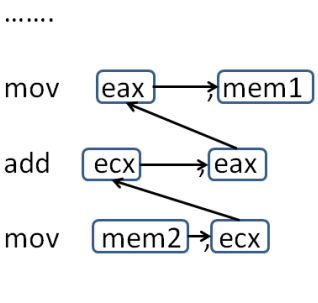

図 8　テイント解析の様子

Figure 8　Flow taint analysis .

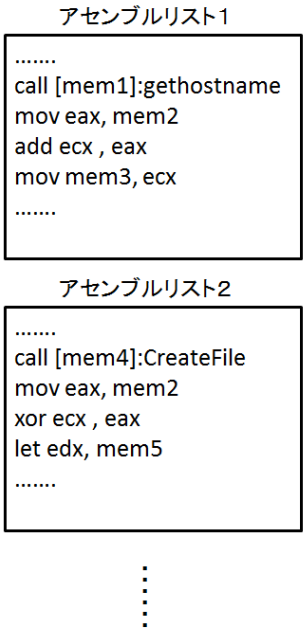

図 9　テイント解析により得られたアセンブルリスト

Figure 9　Assemble lists made by taint analysis.

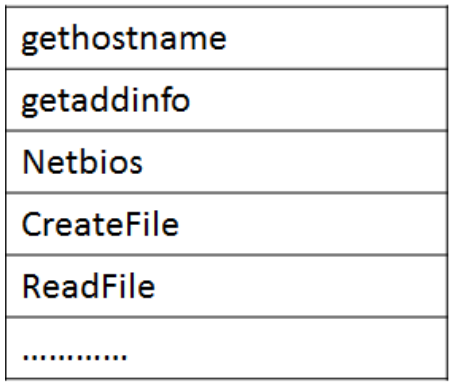

図 10　外部情報参照関数データベース

Figure 10　External information reference function database.

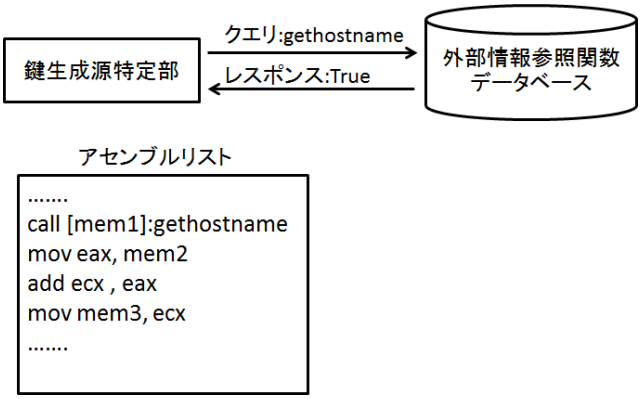

図 11　データベースへの問い合わせの様子

Figure 11　Query for the database.

ここで，外部情報参照関数データベースとは，WinSock のような通信 API や，ファイル読み込みを行う API といった，外部情報を取得するような API が登録されているものである．外部情報参照関数データベースに格納されるデータの例を図 10 に示す．ここで外部情報とは，プログラム内のテーブルといったハードコードされた情報ではないものを指し，IP アドレスや MAC アドレス，時間といった環境毎に変化する情報のことを指す．

図 11 に，複数のアセンブルリストから，どのアセンブルリストが鍵生成源であるかを特定する例を示す．

まず，複数のアセンブルリストから，判定を行うアセンブルリストを取り出す．次に，取り出したアセンブルリストから，call している関数を抽出する．この場合，gethostname がその関数である．次に，gethostname が外部情報参照関数データベースに存在しているかを確認するために，gethostname のクエリを，外部情報参照関数データベースに送信する．外部情報参照関数データベースでは，このクエリが存在しているかを検索する．図 11 の例であれば，gethostname は存在するため，レスポンスとして True を返す．ここで，外部情報参照関数データベースに存在しないクエリを送信した場合，レスポンスとして False を返す．鍵生成源特定部は，True を受け取った時，判定対象であったアセンブルリストが鍵生成源であると判定する．

以上の処理により，マルウェアから，被害鍵を特定するために重要な情報である鍵生成源を自動で得る事ができるようになり，マルウェアによる暗号通信を復号する手間を削減することが可能となる．

5. 考察

5.1 関数の呼び出し情報がない場合

マルウェアが，自作の関数を用いて外部情報源にアクセスする場合や，関数の呼び出し情報がマルウェア作成者により消去されている場合では，関数の情報が外部情報参照関数データベースに登録ができておらず，本手法では鍵生成源を特定できない．そのため，登録する情報を，関数ではなくシステムコールの組み合わせにするといった変更が考えられる．ただし，この場合では実行トレースに加えシステムコールの呼び出しも記録する必要があり，処理が重くなる欠点が考えられる．

5.2 誤検知が多くなる場合

誤検知が増える要因として，テイントの誤伝搬により解析鍵と無関係の情報を検知する事がある．テイントの誤伝搬とは，本来依存関係がなく，追跡しないデータに対して誤ってテイントが伝搬してしまうことをいう．テイントが誤伝搬するような場合を図 12 に示す．

図 12 において，テイント解析を行うと，図 8 と同様に mem2 が mem1 に依存しているという結果が得られる．しかし，「xor　eax, eax」は eax の値によらず eax に 0 を代入する処理である．そのため，実際には mem1 と mem2 の間には依存関係がない．このように，実際には依存関係がないにも関わらず，依存関係があるようにデータがテイントされることをテイントの誤伝搬という．

テイントの誤伝搬が生じることを踏まえると，正確に鍵生成源を特定するためには，誤伝搬により誤った結果を含む鍵生成源候補から，正しい鍵生成源に絞り込む必要がある．

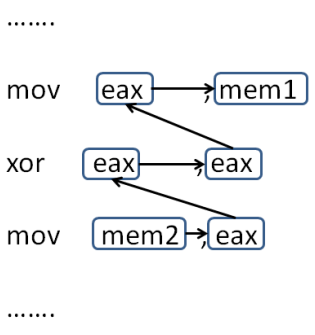

図 12　テイントが誤伝搬する様子

Figure 12　Wrong tainted data propagate.

5.3 被害鍵生成の自動化

本手法により特定した鍵生成源から，被害鍵の生成を自動で行うことができると考えられる．まず，被害環境の情報に仮想環境を設定し，その上でマルウェアを実行する．ここで,マルウェアが暗号処理に利用する鍵を特定すれば，それは被害鍵であるため，より簡単に被害鍵を特定する事が可能になると考えられる．

6. おわりに

マルウェアが，暗号通信で利用する暗号鍵を環境毎に変える場合に，外部情報源を特定する手法を提案した．本手法により，マルウェアを解析した環境と，被害があった環境とで異なる鍵が生成される場合でも，被害環境での鍵を特定する事ができるようになり，マルウェアが行う暗号通信を復号する手間を削減することができる．今後は，本手法を実装し，外部情報源を利用するマルウェアに対して被害鍵が特定できるか実験を行い，本手法の有効性を検証する．

参考文献

[1] 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).

[2] 河内清人, 桜井鐘治: マルウェア通信における暗号化鍵特定手法の提案, コンピュータセキュリティ(CSEC), 2012-CSEC-56(6),1-8

[3] Chi-Keung Luk, Robert Cohn, Robert Muth, Harish Patil, Artur Klauser, Geoff Lowney, Steven Wallace, Vijay Janapa Reddi, Kim Hazelwood. "Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation," Programming Language Design and Implementation (PLDI), Chicago, IL, June 2005, pp. 190-200.