Vol.2016-DPS-166 No.23
Vol.2016-CSEC-72 No.23
2016/3/4

マルウェア解析のためのシステムコールトレースログと通信ログの突合手法

大倉 有喜[1,a)]　大月 勇人[1]　田中 恭之[2]　明田 修平[1]　瀧本 栄二[1]　毛利 公一[1]

概要：マルウェア対策では，感染端末内部での挙動（システムコールや API 呼び出し等）観測や，外部のサーバなどとの通信（ネットワークトラフィック）観測などの手法を用いてマルウェアの挙動を把握する必要がある．従来は，どちらか一方の観測手法を用いることが多かったが，一方の観測結果を解析するだけでは，マルウェアによる詳細な挙動を把握できない．これを解決するために，本論文では，システムコールトレーサ Alkanet のログとネットワークキャプチャのログを用い，両者のログの関連性を明らかにしつつ，両者を紐付けることによってマルウェア全体の挙動を把握可能とするための手法について検討する．

キーワード：動的解析，仮想計算機モニタ，システムコールトレース，パケットキャプチャ

A Method of Combining System Call Trace Log and Captured Packet Data for Malware Analysis

Yuki Okura[1,a)]　Yuto Otsuki[1]　Yasuyuki Tanaka[2]　Shuhei Aketa[1]　Eiji Takimoto[1]　Koichi Mouri[1]

Abstract: Malware analysis uses behavior log or communication log to understand the behavior of malware. Behavior log is acquired by observing system call or API that is invoked by malware inside a computer. Communication log is acquired by observing network communication with C&C servers using packet capturing tool. Previous studies only use either of the observation log. However, analyzing either of the logs is not enough to analyze all malware's behavior. In this paper, we describe the relevance of both logs, and propose a method which combine system call trace log and captured packet data for understanding all behavior of malware.

Keywords: Dynamic Malware Analysis，Virtual Machine Monitor，System Call Trace，Packet Capture

1. はじめに

ボットや RAT に代表されるマルウェアの脅威が高まってきている．シマンテックのレポートによれば，2014 年に新しく発見されたマルウェアの亜種は，3 億 1700 万検体となり，2013 年に新しく発見されたマルウェアの亜種，2 億 5200 万検体よりも大きく増加している [1]．また，ベライゾンのデータ漏洩/侵害調査報告書によれば，マルウェアによるデータ漏洩/侵害事案が年々増加している [2]．これらから，マルウェアの増加とともに企業などの機密情報が狙われていることがわかる．

増加するマルウェアの脅威への対策として，ウイルス対策ソフトウェアによるマルウェアの検出やスパムメールフィルタなどの入口対策，悪性な IP アドレスや URL への通信を遮断するなどの出口対策，マルウェアが接続するサーバの停止など，様々な対策が行われている [3,4]．これらの対策を行うためには，感染端末内部で実行されるマルウェアの挙動や行われる通信，マルウェアと通信する C&C サーバや DNS サーバといった攻撃基盤など，マルウェアに関係する様々な情報を解析することが重要となる．

マルウェアを実行し，その動作を解析する動的解析では，

[1] 立命館大学
Ritsumeikan University，1-1-1 Nojihigashi，Kusatsu，Shiga，525-8577, Japan
[2] NTT コミュニケーションズ株式会社
NTT Communications Corporation，Gran Park Tower 16F, 3-4-1 Shibaura, Minato-ku, Tokyo, 108-8118, Japan
[a)] yookura@asl.cs.ritsumei.ac.jp

ホストベースの観測とネットワークベースの観測の 2 種類の手法で動作の観測が行われている．ホストベースの観測手法は，マルウェアを実行する端末 (以下，実行端末) 内部の挙動を記録する．代表的な挙動観測の手法では，マルウェアに呼び出されるシステムコールや API の記録が行われる．ネットワークベースの観測手法では，パケットキャプチャツールを用いて実行端末によって行われる通信を観測する．

従来の研究では，ホストベースの観測手法とネットワークベースの観測手法の，どちらか一方の観測手法で取得されたログの解析が行われている [5–7]．しかし，ホストベースの観測手法では，行われた通信のすべてを取得することは困難である．パケットのヘッダ情報はカーネルによって処理されるため，TTL といった C&C サーバなどの攻撃基盤の解析に有用な情報 [8, 9] を把握できない問題がある．また，ネットワークベースの観測手法では，実行端末内部の挙動を観測できず，観測した通信が実行端末内部のどのプロセスによって通信が行われているのか把握できない．それぞれの観測手法で観測できない情報は，もう一方の観測手法で観測可能な場合がある．したがって，ホストベースの観測ログとネットワークベースの観測ログを紐付けることで，それぞれのログでは取得できなかった挙動についても取得し解析に利用できると考えられる．さらに，挙動ベースの解析と通信ベースの解析で得られる両者の解析結果を互いに反映しあうことで，両者で悪性な挙動と判定されたことによる解析結果の裏付けや，一方の解析技術では見逃された悪性な挙動の発見が期待できる．

本論文では，感染端末内部での挙動ログと通信ログを突合する手法を提案する．提案手法は，システムコールトレーサ Alkanet [10] を用いて取得したシステムコールトレースログと，パケットキャプチャツールを用いて取得した PCAP ログを利用する．これら両者のログの関連性を明らかにしつつ，両者の紐付けを行うことによって，マルウェア全体の挙動を把握可能とする手法について検討する．

以下，本論文では，2 章で既存研究におけるマルウェア解析手法について述べ，3 章で提案手法について述べる．4 章で Windows 内部の通信処理について述べ，5 章でシステムコールトレースログと通信ログの紐付けについて述べる．6 章で関連研究について述べ，7 章で本論文をまとめる．

2. 既存研究におけるマルウェア解析

本章では，マルウェアを実行した端末内部の挙動に着目した研究と端末によって送受信されるパケットに着目した既存研究について述べる．それぞれの観測手法におけるマルウェア解析について示しつつ，一方の観測結果のみでは解析が不十分であることを示す．

2.1 ホストベースの挙動解析

マルウェアによって行われる典型的な挙動として，プロセス・スレッドの生成，他プロセスへのインジェクション，ファイルやレジストリの操作，通信などが挙げられる．これらの挙動を解析するために，動的解析では実行端末内部でのマルウェアの挙動を，呼び出される API やシステムコールといった粒度で観測し，解析している [5, 6, 10]．

マルウェアは，複数のプロセスの起動や正規のプロセスに対してコードインジェクションを行う．そのため，プロセス・スレッドの生成や，他プロセスへのメモリ書き込みを観測する．

感染を開始したマルウェアは，ファイルやレジストリの読込みや書込みを行い，実行ファイルの作成や実行端末内の機密情報の窃盗，Windows の起動時に自身を自動起動させる設定などを行う．そこで，ファイルやレジストリに対する挙動の観測することで，マルウェアが作成したファイルの名前や作成されたパス，レジストリの変更を取得する．

外部への通信は，2 次検体のダウンロードや C&C サーバからの指令の受信，取得したデータの送信などに用いられる．そのため，通信の挙動を観測することで，送信先 IP アドレス・ポート番号などを取得する．また，送受信されるデータ，すなわち，パケットのペイロードも送受信時に発行される API やシステムコールの引数に格納されているため取得することができる．

2.2 ネットワークベースの挙動解析

ネットワークベースの挙動解析では，マルウェアを実行することで実際に観測された通信を基に解析を行う．通信の観測は，パケットキャプチャツールを用いて行われ，やりとりされるすべての通信のパケットの記録が行われる．通信観測結果は，1 パケットずつ，または，複数のパケットで構成されるコネクション，複数のコネクションで構成されるセッションなどの単位で解析が行われる．特に，マルウェアによって送受信されたデータや通信量の変化，通信先の IP アドレス，利用されたポート番号，DNS のドメイン名などに着目して解析が行われる．

送受信されたデータは，実際にマルウェアによってやり取りされるファイルや C&C サーバからの指令などであり，通信量は，それらがどのような間隔で通信されているのかを示す．特徴的な文字列が送受信されたデータに含まれている場合やマルウェアに見られる通信の間隔などを解析することで，悪性な通信を検出する技術の発展に寄与できる [7, 11]．

マルウェアによって利用される IP アドレスやポート番号，ドメイン名などは，マルウェアがどのサーバとどのような通信をしているのかを把握する上で重要な情報である．これらの情報により，マルウェアが接続する C&C サーバの所在といった情報を把握できる．したがって，解析結果

の結果は，悪性な IP アドレスをブラックリストに登録することによる通信の遮断や，Firewall による特定のポートを利用した通信を禁止といった対策に活用できる．また，マルウェアは，TCP の SYN パケットや DNS パケットを用いて DoS 攻撃を行う．そのため，パケットを解析することで，どのサーバにどのような攻撃が行われていたのかを把握できる．

2.3 ホストベースとネットワークベースの観測手法の課題

ボットネットのような多数の感染端末が連携して動作するマルウェアを解析するためには，実行端末内部での挙動と外部に対して行われる通信，マルウェアの接続するサーバなど，様々な情報を用いて解析することが重要である．しかし，ホストベースとネットワークベースの両者の観測手法の一方のみの観測では，解析に必要な情報が不足するという課題がある．これは，実行端末内部のマルウェアの挙動を解析することや，マルウェアによってどのような通信が行われているのかを解析することを目的に行われているためである．

ホストベースの観測手法では，C&C サーバの情報や C&C サーバまでのネットワーク上の距離など，攻撃基盤を理解する上で有用な情報を取得できない．文献 [8,9] では，TTL などパケットのヘッダに含まれている情報が，攻撃基盤の解析に有用であることが示されている．しかし，パケットのヘッダは，カーネルによって付与・除去が行われるため，API トレースやシステムコールトレースで取得できない．

一方で，ネットワークベースの観測手法によって取得した通信の観測結果を解析するのみでは，実行端末内部のどのプロセス・スレッドに起因する通信であるのかを特定することが困難である．特に，ダウンローダ型のマルウェアは，2 次検体のダウンロード・実行を行う．このとき，2 次検体も通信を行うと，ダウンローダ (1 次検体) と 2 次検体の両方の通信が混交して観測される．このような場合，ダウンロードする 2 次検体が変化すると，通信解析で得られたマルウェアの解析結果が利用できなくなってしまう．また，マルウェアはコードインジェクションにより正規のプロセスに感染するといった動作をするものが存在する．このとき，インジェクションされたプロセスも通信を行う場合，当該プロセス元来の通信とインジェクションされたコードに起因する通信が混在する．

以上から，ホストベースの観測手法では，ネットワークベースの観測手法で観測しているパケットのヘッダ情報を取得できない．また，ネットワークベースの観測手法では，ホストベースの観測手法で観測している実行端末内部のプロセス・スレッドの挙動を取得できない．したがって，両者の観測ログを結びつけ，補いあうことができれば，それぞれの観測手法では不足する情報も取得することできる．

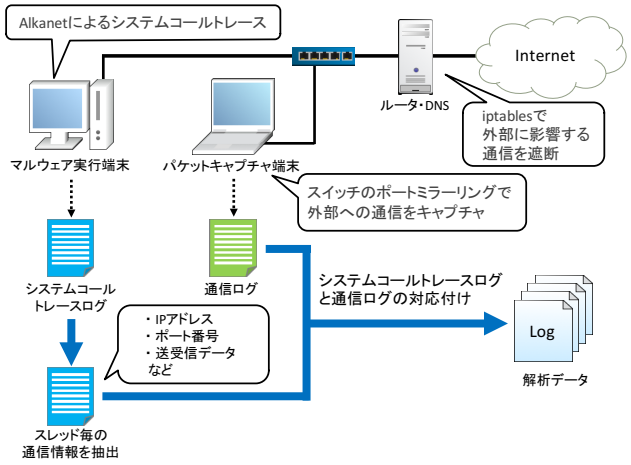

図 1 提案手法の構成と概要

3. 端末内の挙動ログと通信ログの突合

3.1 概要

本論文では，ホストベースの観測手法で取得した実行端末内部の挙動観測ログとネットワークベースの観測手法で取得した通信観測ログを突合する手法を提案する．提案手法では，両者の観測ログに共通して含まれる送信元ポート番号や接続先 IP アドレス・ポート番号などの情報を抽出し，それらの情報に基づき，ログ同士の紐付けを行う．これにより，一方のログを解析するときに，もう一方のログの対応箇所を参照することができる．

実行端末内部の挙動観測では得られなかったパケットのヘッダに含まれる情報が取得できるため，実行端末内部のマルウェアの挙動のみを解析するだけでなく，C&C サーバや他のマルウェア感染端末などについても考慮して解析できるようになる．また，実行端末内部の観測ログで得られた解析結果を通信解析に利用することで，複数のマルウェアによって通信が行われても，それぞれのマルウェアごとに通信を分割して解析することができるようになる．さらに，2 種類の観測ログそれぞれをベースとする悪性挙動判定技術の連携が可能となり，見逃しの防止や判定理由の強化が行える．

3.2 動的解析環境の構成

提案手法では，**図 1** に示す構成によって各ログデータを取得する．実行端末内部のマルウェアの挙動ログは，Alkanet を用いて取得したシステムコールトレースログを用いる．Alkanet は，スレッド単位でマルウェアを識別可能であるため，正規プロセスにインジェクションされたコードによる挙動も観測可能である [10]．実行端末によって行われる通信は，実行端末が接続されたスイッチのポートに対してポートミラーリングを行い，パケットキャプチャ端末でキャプチャする．

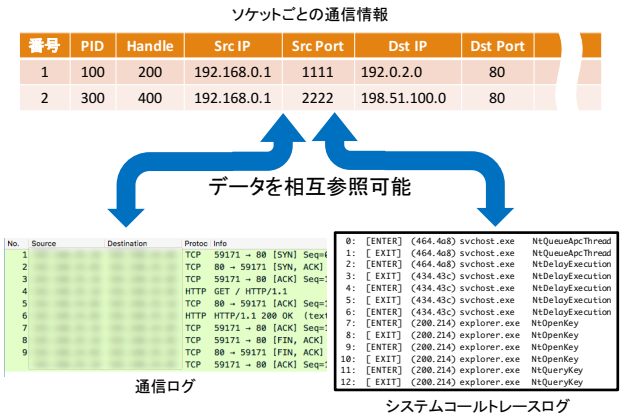

図 2 提案手法によるデータの相互参照

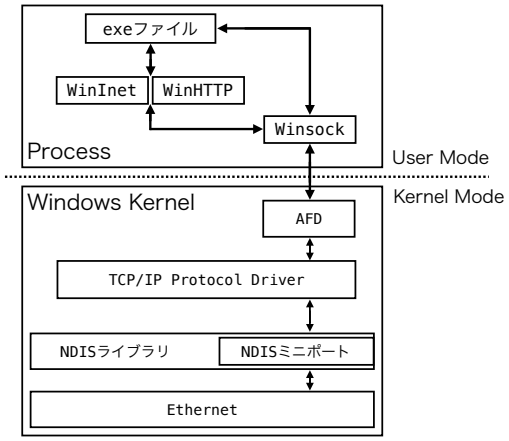

図 3 Windows の送受信処理の流れ

ルータとして動作する PC では，Linux 端末上で iptables と DNS サーバが動作している．iptables では，パケットフィルタリングと IP マスカレードを行っている．実行したマルウェアがグローバルな環境へのスパムメールの送信や DDoS 攻撃などを行う可能性があるため，通信量の制限や送信先のポート番号が Well-Known Port である 0 番から 1023 番ポートについては，DNS，HTTP，HTTPS 以外のポートをすべて閉じている．1024 番ポート以降はすべて開放する．DNS サーバは BIND を用いており，ローカルネットワーク内からの名前解決要求のみ応答する．

Alkanet で取得したシステムコールトレースログから，パケットキャプチャによって取得した通信ログと紐付ける情報を抽出し，両者のログを紐付ける．両者のログを紐付けることにより，両者の観測ログを用いて解析した詳細な解析データが得られる．

3.3 突合手法

提案手法による実行端末内部の観測結果と外部に対して行われる通信の観測結果を突合する手法を**図 2** に示す．提案手法では，システムコールトレースログと通信ログの両方の観測ログを用いて，両者のログに共通する通信挙動に基づく情報を管理するテーブルを作成する．テーブルで管理する情報の詳細については，5 章で述べる．

テーブルは，両方のログから抽出した情報を紐付けることで作成されているため，システムコールトレースログと通信ログの両方のログから参照できる．したがって，実行端末内部の挙動観測ログに記録されている通信挙動で送受信されたパケットは，以下の手順で取得できる．挙動ログに含まれる IP アドレスやポート番号などを用いてテーブルを参照する．テーブルから対応する通信ログが分かるため，実際に送受信されたパケットが取得できる．また，同様にテーブルを参照することで，通信ログのある通信が実行端末内部のどのプロセスのソケットによって行われた通信か判別できる．これにより，そのソケットを用いて通信を行ったスレッドの挙動も取得可能となる．このように，テーブルを用いて相互参照が可能となる．

4. 通信時の Windows 内部の処理

提案手法は，ホストベースの観測手法とネットワークベースの観測手法の両者の観測手法で取得したログから，通信に関する情報を基に紐付けを行う．本章では，Windows 内部での通信を行う処理と実際に送受信されるパケットの関連性について述べる．

4.1 Windows Socket

Windows における代表的な通信 API として，Windows Socket API (Winsock) がある．Winsock は，WinInet や WinHTTP などの通信 API の下位 API となっているため，これら通信 API を使用した場合でも Winsock の処理を必ず経由する．本論文では，Winsock に着目し，Winsock を用いて行われる通信について述べる．

Winsock を用いた通信の流れを**図 3** に示す [12]．Winsock は，AFD (Ancillary Function Driver) を利用して通信を行う．Winsock は通信時に，AFD に制御コードを送信する．制御コードを受け取った AFD は，TCP/IP Protocol Driver に TDI IRP を送信する．TCP/IP Protocol Driver は，IRP 内のデータにプロトコル固有のヘッダを追加し，Ethernet を通して外部へと送信を行う．

Winsock による，AFD への制御コードの送信は，NtDeviceIoControlFile システムコールを用いて実現されている．NtDeviceIoControlFile システムコール [12] は，ファイルハンドルに関連付けられたデバイスドライバに制御コードを送信することで，指定した I/O 制御オペレーションを実行させるシステムコールである．NtDeviceIoControlFile システムコールの定義を**図 4** に示す [13]．NtDeviceIoControlFile システムコールは，FileHandle と IoControlCode の値によって制御が変化する．Winsock の場合は，NtDeviceIoControlFile システムコールの FileHandle が AFD のデバイスファイル

```
NTSTATUS NtDeviceIoControlFile(
  _In_      HANDLE           FileHandle,
  _In_opt_  HANDLE           Event,
  _In_opt_  PIO_APC_ROUTINE  ApcRoutine,
  _In_opt_  PVOID            ApcContext,
  _Out_     PIO_STATUS_BLOCK IoStatusBlock,
  _In_      ULONG            IoControlCode,
  _In_opt_  PVOID            InputBuffer,
  _In_      ULONG            InputBufferLength,
  _Out_opt_ PVOID            OutputBuffer,
  _In_      ULONG            OutputBufferLength
);
```

図 4　NtDeviceIoControlFile システムコールの定義

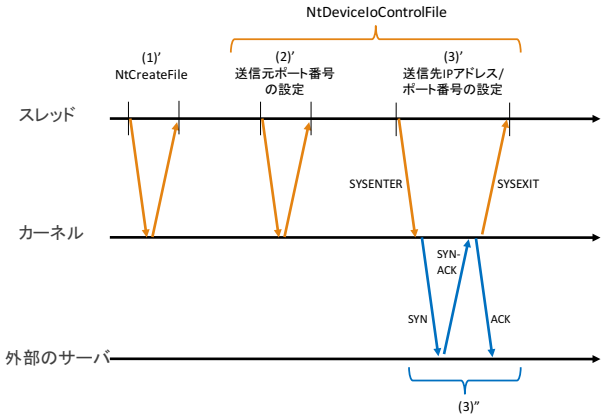

図 5　通信確立時の端末内挙動と通信の関係

を示しており，かつ，IoControlCode が特定の I/O 制御コードを示すときに，送信元ポート番号の設定や送信先 IP アドレス・送信先ポート番号の設定などが行われる．なお，Winsock と AFD でやりとりされる情報は，InputBuffer や OutputBuffer，またこれらに含まれるポインタの先に格納される [14].

4.2　ソケットを用いた通信の流れ

クライアントとサーバの間でソケットを用いた TCP による通信が行われる場合，クライアント側は，主に以下の順に処理を行う．

(1) ソケットを作成する
(2) 送信元ポート番号の設定
(3) 送信先 IP アドレス，送信先ポート番号の設定
(4) データを送信する
(5) データを受信する
(6) ソケットをクローズをする

このときの実行端末内部の挙動と外部に送信されるパケットの関係を**図 5**，**図 6** に示す．図 5，図 6 では，ユーザモードで動作するスレッドと実際にパケットを送信するカーネル，そして，通信先として外部のサーバを示す．

4.2.1　通信確立時の挙動

図 5 は，(1) から (3) の通信を確立するまでの感染端末内部の挙動と外部に送信されるパケットの関係を示す．クライアント側がソケットを用いて TCP の通信を行うとき，まずソケットが作成される．このとき，図中 (1)' の NtCreateFile システムコールが発行され，AFD のファイルオブジェクト \Device\Afd\Endpoint が作成される．このファイルオブジェクトのハンドルは，ユーザモードのスレッドに返される．AFD のファイルオブジェクトは，ソケットへのインタフェースとして扱われる．

次に，作成されたソケットに対して送信元ポート番号や送信先 IP アドレス・ポート番号の設定が行われる．API 内部では，NtDeviceIoControlFile システムコールが発行され，(1)' の NtCreateFile システムコールによって作成されたソケットに対して送信元ポート番号の設定 (2)' や送信先 IP アドレス・ポート番号の設定 (3)' が行われる．このとき，カーネルは，図中 (3)"の 3 ウェイハンドシェイクによるサーバとの接続確立を行う．3 ウェイハンドシェイクの SYN パケットは，(3)' の送信先 IP アドレス・ポート番号の設定を行う NtDeviceIoControlFile システムコールの発行時，すなわち，SYSENTER でカーネルに処理が移った後に送信される．

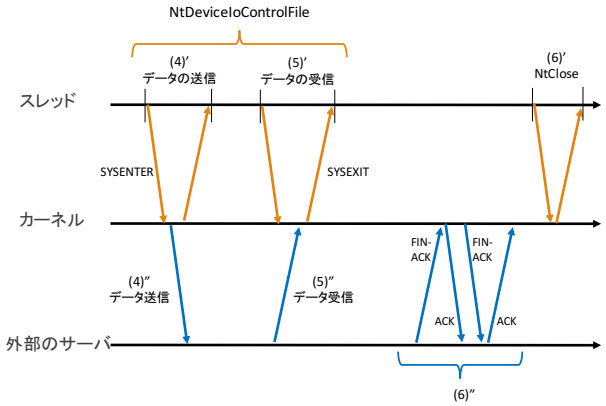

図 6　データ送受信から通信終了までの端末内挙動と通信の関係

4.2.2　データ送受信の挙動

図 6 は，(4) から (6) の通信確立後に行われるデータの送受信から通信の終了までの感染端末内部の挙動と外部に送信されるパケットの関係を示す．このうち，(4) と (5) の挙動について述べる．サーバとの通信が確立した後，サーバへのデータの送信やサーバから送信されてきたデータの受信が行われる．サーバへデータを送信するとき，図中 (4)' の NtDeviceIoControlFile システムコールによりデータをカーネルに渡す．送信されるデータを含んだパケット (4)"は，(4)' の SYSENTER でカーネルに処理が移った後に送信する．

スレッドによるデータ受信は，図中 (5)' のデータを受信する NtDeviceIoControlFile システムコールの発行によって行われる．(5)' の NtDeviceIoControlFile システムコー

ルが発行されるとき，サーバから送信されたデータ (5)” をカーネルが受信していた場合に，スレッドはデータを受信する．このとき，(5)’ の NtDeviceIoControlFile システムコールの SYSEXIT の返り値は，STATUS_SUCCESS である．また，カーネルにデータが届いていない場合は，STATUS_PENDING が返される．

4.2.3 通信終了時の挙動

図 6 の通信終了時の感染端末内部の挙動と外部に送信されるパケットの関係を述べる．ソケットを用いた処理は，(6) の挙動である．通信が終了すると，サーバから FIN-ACK や RST-ACK などの通信を終了するパケットが送信される (6)”．また，スレッドは，図中 (6)’ の NtClose システムコールを発行し，ソケットをクローズする．

通信終了時の実行端末内部の挙動とサーバから送信されるパケットには，時間的関係性が存在しない．これは，通信終了時の挙動に関しては，いつソケットがクローズされるのかはプログラムに依存するためである．通信終了後にクローズするコードが記述されている場合は，そのコードが実行されたときにクローズされる．また，ソケットをクローズするコードが書かれていない場合は，NtClose システムコールは発行されない．

5. ログの紐付け

システムコールトレースログと通信ログは，それぞれの別の挙動観測ツールで取得している．そこで，通信挙動に着目し，両者のログに共通して含まれる情報に基づいて紐付けを行う．本章では，両者のログに共通して含まれる情報について述べ，両者のログの紐付けについて述べる．

5.1 システムコールトレースログに含まれる情報

Alkanet によって取得しているシステムコールトレースログでは，感染端末内部で発行されたシステムコールのうち，マルウェアによる典型的な悪性挙動に利用されるものに絞って記録されている．システムコールトレースログは，発行されたシステムコールの 1 エントリずつに以下の要素を持つ．

- ログ番号
  ログのエントリを一意に識別するための番号
- タイムスタンプ
  Alkanet が起動してからの経過時間
- ログのタイプ
  SYSENTER フック時のログか，SYSEXIT フック時のログかを示す
- システムコール番号
  発行されたシステムコールの番号
- システムコールを発行したプロセス・スレッドの情報
  プロセスの名前，プロセスの識別子，スレッドの識別子
- 引数などの詳細情報
  システムコールの引数がもつ情報など
- 返り値
  システムコールの返り値

4 章で述べたように，Winsock を用いた通信挙動では，主に NtDeviceIoControlFile システムコールが発行される．当該システムコールで設定される送信元ポート番号や送信先 IP アドレス・ポート番号，送受信されるデータは，引数などの詳細情報として記録している．

5.2 通信ログに含まれる情報

ネットワークキャプチャによって取得した通信ログは，PCAP ファイルとして保存される．PCAP ファイルの構成は，グローバルヘッダ，パケットヘッダ，パケットデータ，パケットヘッダ，パケットデータと続く [15]．グローバルヘッダではマジックナンバやバージョンなどが格納されており，パケットヘッダはパケットデータのタイムスタンプやパケット長が格納されている．パケットデータは，実際に送受信されたパケット本体である．

TCP パケットの場合，パケットデータは，イーサネットヘッダ，IP ヘッダ，TCP ヘッダ，ペイロードの構成となる．また，UDP パケットの場合，パケットデータは，イーサネットヘッダ，IP ヘッダ，UDP ヘッダ，ペイロードの構成となる．IP ヘッダには，TTL や送信元・送信先の IP アドレスなどが格納されている．TCP ヘッダ・UDP ヘッダには，利用した TCP・UDP のポート番号などが格納されている．ペイロードは，送受信されたデータである．

5.3 システムコールトレースログと通信ログの紐づく情報

Alkanet によるシステムコールトレースで取得したログとパケットキャプチャで取得した PCAP ファイルで共通して含まれる情報として，以下が挙げられる．

- タイムスタンプ
- 送信元ポート番号
- 送信先 IP アドレス
- 送信先ポート番号
- 送信・受信データ

パケットデータが TCP パケット，または，UDP パケットの場合，パケットデータには，送信元ポート番号と送信先 IP アドレス・ポート番号などが含まれている．Alkanet でも，ソケットに紐付いた送信元ポート番号や送信先 IP アドレス・ポート番号などを取得している．また，それぞれの観測結果には，データを取得したタイムスタンプが含まれている．PCAP ファイルでは，ソケットごとの通信の開始から終了までの期間を取得することができる．Alkanet では，ソケットごとの通信の開始時間が取得できる．

これらの共通して含まれる情報を基に，図 2 の上部で示したソケットごとに通信を結びつけるテーブルを作成する．テーブルに含まれる情報の一覧を以下に示す．

- ソケットを識別する番号
- 送信元 IP アドレス・ポート番号
- 送信先 IP アドレス・ポート番号
- プロセスの識別子
- ファイルハンドル
- PCAP ファイルに含まれる通信の開始時間
- PCAP ファイルに含まれる通信の終了時間
- システムコールトレースログに含まれる通信の開始時間

システムコールトレースログと通信ログに共通して含まれる情報の他に，それぞれの通信を区別するために追加すべき情報が存在する．4 章で述べたように，AFD のファイルオブジェクトは，ソケットへのインタフェースとして扱われる．また，AFD のファイルオブジェクトへのアクセスは，ファイルハンドルを用いる．したがって，ソケットとファイルハンドルは，ファイルオブジェクトを介して 1 対 1 で対応する．すなわち，実行端末内部では，ファイルハンドルを利用してそれぞれの通信が区別できる．実行端末内部の挙動との結びつけを容易にするために，テーブルにファイルハンドルも追加する．また，ファイルハンドルは，プロセスごとにハンドルテーブルで管理されている．どのプロセスの通信なのかを区別するために，プロセスの識別子 (PID) もテーブルに追加する．PCAP ファイルは，それぞれの TCP・UDP パケットに送信元 IP アドレスが含まれる．現行の Alkanet のシステムコールトレースログには，送信元 IP アドレスが含まれていない．しかし，送信元 IP アドレスは，通信を識別する上で重要な情報であるため，テーブルに追加する．

作成したテーブルによって，PCAP ファイルの観測した通信から，その通信に紐付く実行端末内部のファイルハンドルと PID を取得できる．ファイルハンドルと PID から通信を行ったスレッドを特定でき，当該スレッドによって発行された他のシステムコールも取得し解析できる．また，実行端末内部のスレッドによって行われた通信の，実際に観測されたパケットデータを PCAP ファイルから取得することができるため，サーバからのレスポンスパケットのヘッダに含まれる TTL などを取得でき，サーバの情報を解析に反映できる．

6. 関連研究

三村 [16] らと神薗 [17] らは，ホストベースの観測手法とネットワークベースの観測手法を併用することで，悪性な通信を行うプロセスの特定をしている．三村らは，Windows のカーネルモードドライバによって，通信を行うプロセスを特定する手法を提案している．ドライバでは，観測したすべてのプロセスについて，起動・終了・モジュールの読み込み・通信試行の 4 項目を記録する．これにより，不正な通信を検知した際に記録したログと比較することで，不正な通信を行ったプロセスを特定・解析することができる．神薗らは，標的型攻撃対策として，Windows API フックを用いることで，起動したプロセス及びすでに起動しているプロセスの一連の通信手続きを保全するフォレンジック手法を提案している．神薗らのシステムで出力されたログは，プロセス情報と通信内容の 2 つの情報を持っている．ホストベース監視製品でアラートが発生したときに，ログのプロセス情報を突合することでどのような通信が行われていたかを判別することができる．また，ネットワーク監視製品でアラートが発生したとき，ログの通信内容と突合することでどのプロセスによって通信が行われていたかを特定することができる．

これらは提案手法と目的が異なるが，通信ログからその通信を行ったプロセスを特定可能としていることに類似性がある．既存研究は，通信を行ったプロセスの特定までを行うのに対し，提案手法は，通信挙動を起点として，ソケットごとに通信の紐付けを行う．したがって，プロセス単位だけでなく，スレッド単位で通信との紐付けも行うことが可能である．これにより，通信を行ったスレッドによって行われた挙動も含めた解析を可能としている．

Fan [18] らは，実行端末内部の挙動に着目し，機密データを流出させるソフトウェアの発見を行っている．この研究では，API トレースとシステムコールトレースを用いて，実行端末内部のプロセスの状態遷移を観測することで挙動と通信の関係を明らかにしている．観測結果から，機密データを読み出す処理フローと，通信により外部へ機密データを出力する処理フローを作成する．これらの処理が重なる場合，外部へと機密データを送信するソフトウェアとみなす．

提案手法もシステムコールトレースにより，実行端末内部のプロセス・スレッドの通信時に発行されるシステムコールに着目して解析を行っている．一方で，実行端末内部の挙動だけでなく，実際に観測されたパケットも用いて相互に情報を補完する点が Fan らの研究と異なる．

7. おわりに

本論文では，マルウェア全体の挙動を把握するために，端末内部での挙動のログと実行端末によって行われる通信のログを突合する手法について述べた．提案手法では，端末内部の挙動ログとしてシステムコールトレースログを用い，実行端末によって行われる通信のログとして PCAP を用いる．これら両者のログの関連付けることで両者のログを紐付け，一方の観測結果のみを解析するだけでは把握できないマルウェアの挙動を解析する．今後は，ソケットごとの通信を紐付けるテーブルだけでなく，同じ時間軸で両者のログを参照できる，または，1 パケットごと，1 システムコールごとに互いのログを参照できるなど，様々な目的で両者のログを参照可能とするデータ構造の設計とシステ

ムの実装を行う．

参考文献

[1] Symantec: 2015 年インターネットセキュリティ脅威レポート第 20 号 (2015).
[2] verizon: 2014 年度 データ漏洩/侵害調査報告書 (2014).
[3] 独立行政法人 情報処理推進機構セキュリティセンター：新しいタイプの攻撃」の対策に向けた設計・運用ガイド, 独立行政法人情報処理推進機構セキュリティセンター（オンライン），入手先 ⟨https://www.ipa.go.jp/files/000017308.pdf⟩（参照 2016-02-03）.
[4] 警察庁：国際的なボットネットのテイクダウン作戦，警察庁（オンライン），入手先 ⟨https://www.npa.go.jp/cyber/goz/⟩（参照 2016-02-06）.
[5] Carsten, W., Holz, T. and Freiling, F.: Toward Automated Dynamic Malware Analysis Using CWSandbox, *IEEE Security and Privacy*, Vol. 5, No. 2, pp. 32–39 (2007).
[6] Bayer, U., Kruegel, C. and Kirda, E.: TTAnalyze: A Tool for Analyzing Malware, *15th European Institute for Computer Antivirus Research (EICAR 2006) Annual Conference* (2006).
[7] Gu, G., Perdisci, R., Zhang, J. and Lee, W.: BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-independent Botnet Detection, *Proceedings of the 17th Conference on Security Symposium*, SS'08, USENIX Association, pp. 139–154 (2008).
[8] 東角芳樹，寺田剛陽，鳥居　悟：ネットワーク上の距離に着目したマルウエアの配置に関する一考察，コンピュータセキュリティシンポジウム 2010 論文集，Vol. 2010, No. 9, pp. 891–896 (2010).
[9] Ryo Yamada and Shigeki Goto: Using abnormal TTL values to detect malicious IP packets, Vol. 34, pp. 27–34 (2013).
[10] 大月勇人，瀧本栄二，齋藤彰一，毛利公一："マルウェア観測のための仮想計算機モニタを用いたシステムコールトレース手法"，情報処理学会論文誌，Vol. 55, No. 9, pp. 2034–2046 (2014).
[11] Goebel, J. and Holz, T.: Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation, *Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets*, HotBots'07 (2007).
[12] Solomon, D. and Russinovich, M.: インサイド Microsoft Windows 第 4 版 下，pp. 365–428, 日経 BP ソフトプレス (2008).
[13] Microsoft Developer Network: NtDeviceIoControlFile function (Windows), Microsoft Corporation (online), available from ⟨https://msdn.microsoft.com/en-us/library/ms648411(v=vs.85).aspx⟩ (accessed 2016-02-03).
[14] Helmut Petritshch: *Network Virtualisation*, pp. 39–51, VDM Verlag Dr. Mueller (2008).
[15] The Wireshark Wiki: Libpcap File Format, The Wireshark Wiki (online), available from ⟨https://wiki.wireshark.org/Development/LibpcapFileFormat⟩ (accessed 2016-02-03).
[16] 三村聡志, 佐々木良一：プロセス情報と関連づけたパケットを利用した不正通信原因推定手法の提案，マルチメディア、分散協調とモバイルシンポジウム 2014 論文集, Vol. 2014, pp. 1973–1980 (2014).
[17] 神薗雅紀，遠峰隆史，衛藤　侑，星澤裕二，井上大介：プロセスの通信手続きに基づくフォレンジック手法の提案，コンピュータセキュリティシンポジウム 2014 論文集，Vol. 2014, No. 2, pp. 167–174 (2014).
[18] Fan, L., Wang, Y., Li, J., Cheng, X. and Lin, C.: "Privacy Petri Net and Privacy Leak Software", *Jounal of Computer Science and Technology*, Vol. 30, No. 6, pp. 1318–1343 (2015).