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

神薗 雅紀[†1†2]　遠峰 隆史[†1]　津田 侑[†1]　衛藤 将史[†1]　星澤 裕二[†2]　井上 大介[†1]

†1 独立行政法人 情報通信研究機構 〒184-8795 東京都小金井市貫井北町 4-2-1
†2 株式会社セキュアブレイン 〒102-0083 東京都千代田区麹町 2-6-7 麹町 RK ビル 4F
E-mail: †1{masaki_kamizono, tomine, tsuda, eto, dai}@nict.go.jp
†2 {masaki_kamizono, yuji_hoshizawa}@securebrain.co.jp

あらまし　標的型攻撃対策として様々なセキュリティ製品が登場している．何らかのインシデントによりネットワーク監視製品がアラートを挙げた際は，アラート情報を基に対象ホスト内のどのプロセスによる通信であるかツール等を用いて解析する．しかし，これらのツールは主にプロセスの状況や通信先情報等を提供するのみであり，アラート情報と突合するために必要となるプロセスの通信手続き情報が保全されておらず，分析に多くの時間を要し，かつ推測の入った解析結果となるケースが多い．そこで本稿では，プロセスの一連の通信手続きを保全することでプロセスと通信を結び付け，セキュリティ製品のアラートと突合することによるフォレンジック手法を提案する．さらに，プロセスの通信手続きの一つである DNS クエリに着目し，保全情報から不正なプロセスを特定する手法を考察する．

キーワード　標的型攻撃，フォレンジック，ログ保全，プロセス通信，DNS

Proposal of Forensics Method Based on Communication Procedure of Process

Masaki KAMIZONO[†1†2]　Takashi TOMINE[†1]　Yu TSUDA[†1]　Masashi ETO[†1]
Yuji HOSHIZAWA[†2]　and　Daisuke INOUE[†1]

† 1 National Institute of Information and Communications Technology
4-2-1 Nukui-Kitamachi, Koganei, Tokyo, 184-8795 Japan
† 2 Securebrain Corporation Kojimachi RK Bldg, 6-7 Kojimachi 2-chome, Chiyodaku, Tokyo, 102-0083 Japan
E-mail: †1{masaki_kamizono, tomine, tsuda, eto, dai}@nict.go.jp
†2 {masaki_kamizono, yuji_hoshizawa}@securebrain.co.jp

Abstract In common digital forensic operations using network security appliances against APT, based on the alerts published by the appliances, operators try to find out processes that conducted suspicious communications and consequently caused the alerts. However, the conventional tools used in the operation only provide information such as status of processes and destinations of the communications, that are insufficient to obtain conclusive evidence for practical cases. Since these tools do not preserve and provide communications procedure of each process, which is crucial to correlate network activities and alerts reported by network appliances, forensic operations take longer time and the results must have a certain ambiguity. In order to address this problem, we propose a forensic method which enables to correlate processes and alerts by preserving communication procedures of internal processes of hosts. Particularly, this paper focuses on DNS query, which is an important step of a communication procedure, and presents considerations of how to specify malicious processes based on the preserved information.

Keyword　Advanced Persistent Threat，Digital Forensics，Log Preservation，Communication Procedure，DNS

1. はじめに

マルウェアを利用したサイバー攻撃による被害が多発しており，中でも特定の組織や企業を標的とし，機密情報の収奪や特定システムの破壊を目的とした標的型攻撃は，会社や組織間さらには国家をまたぐ問題となっている．このような背景の下，標的型攻撃対策として次世代ファイアウォールや侵入検知装置等で代表されるネットワーク監視製品，アンチウイルスソフトなどで代表されるホストベース監視製品など，様々なセキュリティ製品が登場しており，インシデントが発生した際は，各種セキュリティ製品のアラート情報やログ情報を基にフォレンジックを行うことで，原因の究明や対策方法を立案する．しかし，フォレンジックを行った際に，その結果にある程度の推測が入ることや，分析に多くの時間やコストを要することから，効果的なフォレンジック手法が希求されている．

フォレンジックに推測が入ることや多くの時間を要する要因として，本来必要なログ情報が保全されていないこと，各種セキュリティ製品のアラート情報やログ情報がお互いに連携や突合できないことが挙げられる．例として，ネットワーク監視製品が外向きの通信に対してアラートを発生した場合を挙げる．まず，

アラートよりどの端末からの通信であるか特定し，さらに端末内のどのプロセスが当該アラートの原因となった通信を発生させたのか，その実行ファイルはどこに保存されているか分析していく．しかし，ネットワーク監視製品のアラートやログ情報は基本的には通信レイヤーにおける情報であるため，当該情報だけでは端末内部のどのプロセスによる通信であるか等を判別することは容易ではない．このため，一般的にホストベース製品や端末のログ，netstat コマンドなど OS の標準コマンドを利用して得られた情報，さらには端末にプロセスの挙動を収集するツール等を導入して得られる分析時における端末の“通信先の IP アドレスや逆引きされたホスト名”等の情報を利用し，ネットワーク監視製品のアラートに関係するプロセス等を特定もしくは推測する．従って，分析時に得られた端末側の情報とネットワーク監視製品のアラートが一致した場合は，不正なプロセス等を比較的容易に特定することができるが，インシデントが発生し時間が経過した場合は,情報が保全されていないため分析が困難となる．さらに，昨今のネットワーク監視製品は単純にシグネチャ，通信先 URL および IP アドレス情報のみではなく，DNS やレピュレーション情報などの様々な情報を利用して不正な通信に対しアラートを挙げることで，その精度を向上させているものが多く存在する．これゆえ，端末側で得られる“通信先の IP アドレスや逆引きされたホスト名”といった情報では，昨今のネットワーク監視製品によるアラート情報と突合することが難しい．効果的にフォレンジックを行うためには，これらのネットワーク監視製品が挙げたアラート情報と突合するために，過去から現状の通信状況に至るまでのプロセスの通信手続きにおける情報を保全しておく必要がある．具体例として Web ブラウザを用いて Web サイトを閲覧する場合を挙げると，DNS クエリを発行後，HTTP リクエストを行い，データを取得する．このような一連の処理をプロセスの通信手続きとする．現状では上記に示すようなプロセスの通信手続きが保全されていないため，アラートが発生した時刻における端末のログを分析するといった，ある程度の推測の入った分析結果となり，同時に時間を要する作業となる．

続いて，ホストベース監視製品がアラートを挙げた場合を例に挙げる．この場合は，どのファイルがマルウェアであったのか容易に判別できるケースが多い．しかし逆に，そのマルウェアの起動プロセスがどのような通信を行っていたか，そのアラートからでは判断が困難であり，さらにアラート情報からネットワーク監視製品等のログとも突合することが困難である．このため，例えばホストベース製品のアラート情報を基にネットワーク監視製品等のログと突合することで，インシデントに関連する IP アドレスや URL，ポート番号や DNS 情報などを導出し，その結果をネットワーク監視製品に反映させて不正な通信をブロックするといった対策に反映させることが難しい．

そこで本稿では，プロセスの一連の通信手続きを保全することで，ネットワークおよびホストベース監視製品のアラート情報やログ情報を突合する効果的なフォレンジック手法を提案する．特に，近年のネットワーク監視製品は DNS 情報を用いたアラートを挙げるものが台頭していることも踏まえ，今回はプロセスの通信手続きの一つである DNS クエリに着目し，提案手法の有効性を示す．

本論文の構成は以下のとおりである．まず 2 章にて関連研究について述べ．続いて 3 章にて提案手法を説明する．その後，4 章にて提案手法を用いた検証実験を述べる．5 章にて考察を行い，最後にまとめと今後の課題とする．

2. 関連研究

ネットワーク監視機器（IDS 等）のアラート情報から端末内の正規のプロセスに成り済ます不正なプロセスを特定する手法として，山本[1][2]らの手法が提案されている．山本らの手法は IDS 等のアラート情報である送信元ポート番号をトリガとし，端末の netstat コマンドを利用して得られた解放ポート番号とアラート内容を突合することで関連プロセスを特定する．次に，特定されたプロセスの情報を解析システムが分析し，コード注入等が確認された場合は不正なプロセスであったと判定する．

三村[3]らの手法は，カーネルドライバにて端末に導入し,OS 起動時からのプロセスとその通信のログをとり続けるシステムを開発している．OS 起動時からのプロセスの通信ログを取り続けることで，何らかのインシデントが発生した際に，過去の情報を遡り分析可能としている．ログに記録される情報は，プロセス ID，親プロセス ID，実行イメージファイルパス，接続元・接続先 IP アドレス/ポート番号などである．

その他，関連するコマンドとして Windows OS の netstat コマンド，ツールとしては Wireshark[4], Process Monitor[5], Process Hacker[6]などが挙げられる．netstat コマンドは通信中の TCP コネクションの状態を表示させ，ESTABLISHED となっている通信に関しては，通信先の IP アドレスや逆引きされたホスト名を表示する．また，Wireshark はパケットアナライザソフトウェアであり，パケットの内容を分析・表示することが可能であるが，どのプロセスによるパケットであるか判定できない．Process Monitor，Process Hacker は起動

プロセスの詳細な情報を収集可能であるが，netstat と同様にプロセスの通信手続き情報は取得できない．

以上より，三村らの手法および OS のコマンドやプロセス状況を監視するツールは，接続元・接続先 IP アドレス/ポート番号，接続先 IP アドレスからの逆引きされたホスト名等を収集・保全しているが，フォレンジックの際に重要となる現状の通信状態に至るまでのプロセスの通信手続きは全く保全しない．また，山本らの手法は IDS のアラートと，端末内の netstat コマンド実行結果を突合している．これより， netstat の情報と突合できるものに対して有効な手法である．

関連研究および関連ツールと本稿における提案の明確な差異は，今までの研究やツールでは保全対象とされていなかったプロセスの一連の通信手続きをログファイルとして保全することでプロセスと通信を関連付け，さらに保全情報を用いて効果的にフォレンジックを行う点である．

3. 提案手法

本節では，プロセスの通信手続き情報の保全手法，ならびに，保全した情報を利用したフォレンジック手法を述べる．

3.1. プロセスの通信手続き情報の保全手法

プロセスが起動され通信を行った際の通信手続き保全概要を図 1 に示す．図 1 に示す通り，提案手法では起動するプロセスおよび既に起動されたプロセスに対し，Windows API フック処理を仕掛けることでプロセスの通信手続きを保全する．netstat コマンドやプロセス監視ツールでは，現状の開放ポートなどの通信先の IP アドレスや逆引きホスト名などを取得するが，提案手法ではプロセスの起動時から現状の通信状態に至るまでの DNS，HTTP および送受信処理（TCP/UDP）などの一連の通信手続きを保全する．

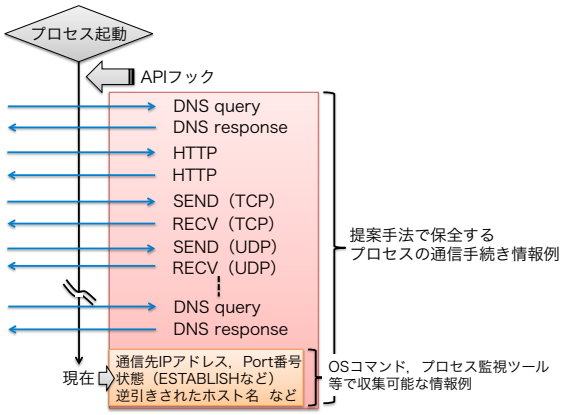

図 1 プロセスの通信手続き保全概要

本稿では，Windows API フックを用いてプロセスの通信手続き情報を保全する．以下，プロセスの通信手続きを保全する具体的な手法ならびに保全形式を示す．

3.1.1. Windows API フックを用いたプロセスの通信手続き保全手法

本稿では，Windows API フックを用いて，プロセスの通信手続き情報を保全する．起動するプロセスおよび既に起動されたプロセスに対し，Windows API フック処理を仕掛け，通信手続き情報をプロセス毎に CSV 形式のファイルとして保存する．また，当該プロセスの他（子）プロセスの起動や，他のプロセスへインジェクションした場合，それらのプロセスに対しても自動的に同様の Windows API フック処理を設け，別プロセスとして通信手続きを保全する．なお，既に起動されているプロセスに対してフック処理を仕掛けた場合は，フック処理後の情報のみ保全可能となる．

フックするプロセスの通信手続きに関する Windows API ならびにその概要の一例を表 1 に示す．

表 1 フックする Windows API 例

DLL/API	概要
kernel32.dll	–
LoadLibraryExW	ws2_32.dll,wsock32.dll,dnsapi.dll がロードされた場合にフックを設定する
CreateProcess	新しく起動されたプロセスにフックを設定する
CreateRemoteThread	インジェクションした他のプロセスをフック対象とする
dnsapi.dll	–
DnsQuery_A/UTF8/W	DNS のログを出力
wsock32.dll	–
recv，send	通信ログを出力
recvfrom，sendto	通信ログを出力．また，データが DNS のリクエスト/レスポンス形式の場合，内容を解釈しログを出力
gethostbyname	DNS のログを出力
ws2_32.dll	–
recv，send	通信ログを出力（TCP）
recvfrom，sendto	通信ログを出力（UDP）
gethostbyname	DNS のログを出力
GetAddrInfoW	DNS のログを出力
getaddrinfo	DNS のログを出力

なお，表 1 の sendto や recvfrom API を利用して直接 DNS 通信を行うものについは，通信内容が DNS のプロトコル形式に合致するか判定し，解釈後にログを出力する．DNS フォーマットは RFC 1035[7]に従う．

3.1.2. 保全形式ならびに保全内容

本節では，3.1.1 節で述べた手法にて取得した情報の保全フォーマットを示す．提案システムでは，プロセスの通信手続きを CSV 形式として保全する．各行における保全フォーマットを図 2 に示す．

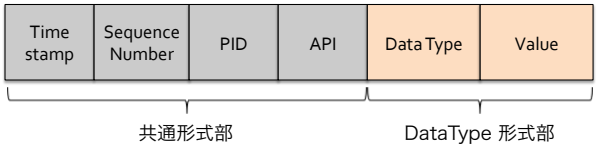

図 2 保全フォーマット

続いて，図 2 に示す共通形式部における各項目の概要を表 2 に示し，DataType 形式部を表 3 に示す．DataType は，タイプ毎に Value 値が異なる．

表 2 保全フォーマット（共通形式部）

分類	概要
Timestamp	ログを出力した時間となり，yyyy/mm/dd hh:mm:ss.mmm 表記とする
Sequence Number	1 回の API フックごとに 1 ずつ増加するシーケンス値
PID	当該プロセス ID
API	実行された API 名であり，DLL 名_API 名表記とする

表 3 保全フォーマット（DataType 形式部）

Data Type	Value
PID	対象プロセスの情報となる以下の情報を記録する． 共通形式,PID,プロセス ID,実行ファイルパス ※ログの 1 行目に 1 回のみ出力
PPID	対象プロセスの親プロセス情報として以下の情報を記録する． 共通形式,PPID,親プロセス ID,親プロセス実行ファイルパス ※ログの 2 行目に 1 回のみ出力
NAME	API（DnsQuery_A，DnsQuery_W，DnsQuery_UTF8，GetAddrInfoW，GetAddrInfoW, getaddrinfo, gethostbyname）によってドメイン名からアドレスを取得した際に，その内容を以下の形式にて記録する． 共通形式,NAME,ドメイン名,IP アドレス,...(IP アドレスが複数返った場合)
SEND, RECV	ネットワークにデータを送受信したことを表す．(TCP) ※データは文字列として解釈できる場合のみ，最大 256 文字，改行コードまでの 1 行が出力 共通形式,SEND/RECV,ローカル IP アドレス:ポート,リモート IP アドレス:ポート,送受信データサイズ,"送受信データ"
SENDTO, RECVFROM	ネットワークにデータを送受信したことを表す．(UDP) ※形式は SEND,RECV と同様 ※SENDTO,RECVFROM の場合，送信データが DNS のリクエスト，レスポンス形式に一致した場合，次の行に DNSQ あるいは DNSA のデータを出力．（シーケンス番号を同じ値とする）
DNSQ DNSA	DNS クエリ/レスポンスであることを表す 共通形式,DNSQ/DNSA,問い合わせの ID,ドメイン名,...(ドメイン名が複数の場合)

3.2. 保全情報を利用したフォレンジック手法

本節では，プロセスの通信手続き情報を利用したフォレンジック手法を述べる．提案ツールを利用した際，プロセス毎に 3.1.2 節に述べた内容の保全情報が得られる．これらの情報とネットワークおよびホストベース監視製品等のアラート情報やログ情報と突合することで，効果的なフォレンジックを実現する．フォレンジック概要を図 3 に示す．

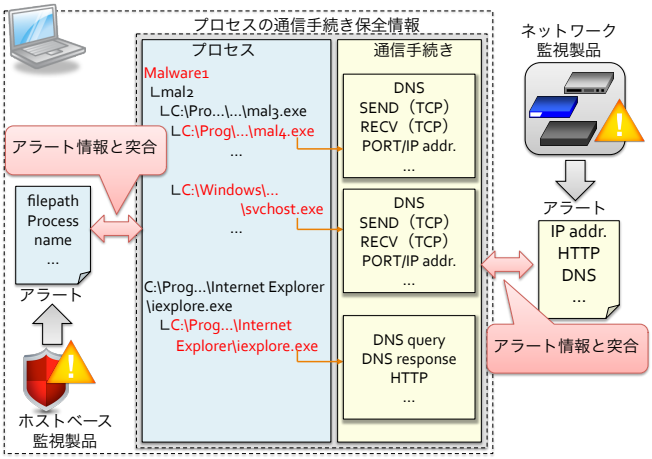

図 3 プロセスの通信手続きを利用したフォレンジック概要

提案ツールの保全情報は，プロセスの親子などの相関関係と，各プロセスの通信手続き情報を保全している．つまり，各プロセスとその通信を結び付けていることになる．これより，例えばネットワーク監視製品が何らかのアラートを発行した際，各プロセスの通信手続き情報と突合することが可能となる．突合が成功した際は，自動的にどのプロセスによる通信であったか把握できることになり，さらに PID や実行ファイルパス，PPID 等も把握できるため，これらの情報を利用することで容易に感染源のプロセスや関連プロセス等を特定することも可能となる．

また，ホストベース監視製品によるアラートが発行

された場合も同様，保全情報に含まれるプロセスの実行ファイルパス等の情報と突合することが可能であり，先程と同様に PPID 情報を用いて感染源まで特定することが可能となる．さらに，関連するプロセスの通信手続き情報を利用することで，接続した IP アドレスや利用ポート，さらにはどのような DNS クエリや HTTP アクセスを行い，どのような処理を行ったか把握することができ，これらの情報をネットワーク監視製品等にフィードバックすることが可能となる．

ただし，セキュリティ製品のアラートやログ形式は独自のフォーマットが利用されている場合が多く，これらの情報と自動的に突合するためには，製品に合わせた情報の抽出処理を導入する必要がある．本稿では，これらの処理の開発は対象外としている．

4. 検証実験

本章では，実際のマルウェアに対して提案手法を用いた検証実験を行う．最初に，様々な通信を発生させる独自プログラム，正常プログラムとして Internet Explore（iexplore.exe），Mozilla Firefox（Firefox.exe）を利用して Web アクセスを行い，プロセスの通信手続きが正確に保全できていることを確認する．続いて，実際のマルウェアに対してプロセスの通信手続き情報が保全できるか検証後，セキュリティ製品のアラートが発生したと仮定し，提案手法であるプロセスの通信手続きに基づくフォレンジック手法の有効性を示す．

4.1. 検証環境

検証環境を表 4 に示す．また，マルウェアを利用した検証は，感染後約 5 分間の通信手続き情報を保全する．なお，提案ツールが仕様通りプロセスの通信手続き情報を保全できているか確認するため，Host OS 側にてパケットキャプチャし，開発ツールの保全情報と比較することで検証する．

表 4 検証環境

検証環境スペック	
CPU	Intel Core i7 2GHz
メモリ	8GB
HOST OS	OS X 10.9.3
Guest OS	Windows 7 Professional 32bit（UAC off）

4.2. 正常アプリケーションでの検証

本節では，正常アプリケーションを用いて開発ツールの動作検証を行う．具体的には，表 5 に示す検証手法にて，期待通りプロセスの通信手続きが保全されていることを確認する．なお，最終的な確認は，提案システムが保全した情報と Wireshark にて取得した情報とを比較することで検証する．

表 5 に示す検証結果のとおり，正常なアプリケーションにおいては，提案ツールは期待通り情報を保全することができた．

表 5 検証プログラム

検証プログラム	検証方法	検証結果
独自プログラム	以下の API を発行させ検証 DnsQuery_A/W/UTF8 GetAddrInfoW，getaddrinfo Gethostbyname DNS(SENDTO,RECVFROM) CreateRemoteThread(iexplore.exe へインジェクション)	○
iexplore.exe (v 8.0.76)	Web サイトへアクセスし検証 (http://www.yahoo.co.jp/ など)	○
Firefox.exe(v 31)		○

4.3. マルウェアによる検証

本節では，実際のマルウェアに対して提案ツールを用いてプロセスの通信手続きを保全できるか検証し，セキュリティ製品のアラートと突合することでどのようにフォレンジックが可能であるか述べる．

4.3.1. 検体 1

検体 1 である W32.Ramnit.B!inf[8]（Symantec 社名）を実行し，提案ツールにて保全したプロセスの通信手続き情報の抜粋を図 5～図 9 に示す．なお，以降の保全情報は省略のため Timestamp の yyyy/mm/dd 部を削除している．保全情報にはプロセスの親プロセス情報が保全されているため，当該情報を利用したプロセスの相関関係を図 4 に示す．検体 1 から複数のプロセスが起動されていることが分かる．

続いて，各プロセスの通信手続きを保全情報である図 5～図 9 を分析する．

```
検体 1(PID: 3424)
 └ f..028mgr.exe(PID: 3440)
   └ C:¥Program Files¥Microsoft¥WaterMark.exe(PID: 3468)
     └ C:¥Program Files¥Microsoft¥WaterMarkmgr.exe
     └ C:¥Windows¥system32¥svchost.exe(PID: 3512)
     └ C:¥Windows¥system32¥svchost.exe
 └ C:¥Program Files¥Microsoft¥WaterMark.exe(PID: 3460)
   └ C:¥Program Files¥Microsoft¥WaterMarkmgr.exe
   └ C:¥Windows¥system32¥svchost.exe
   └ C:¥Windows¥system32¥svchost.exe
```

図 4 プロセスの相関関係

```
01:57:23.202,1,3424,-,PID,3424,C:¥malware¥ 検体 1.exe
…
01:57:23.217,2,3424,kernel32_CreateProcess,PROCESS,(
null),C:¥malware¥ f..028mgr.exe
```

```
01:57:23.248,3,3424,kernel32_CreateProcess,PROCESS,(
null),C:¥Program Files¥Microsoft¥WaterMark.exe
```

図 5 保全情報（検体 1.exe PID:3424）

```
01:57:23.248,1,3440,-,PID,3440,C:¥malware¥ f..028mgr.e
xe
01:57:23.248,1,3440,-,PPID,3424,C:¥malware¥ f..028.exe
01:57:23.264,2,3440,kernel32_CreateProcess,PROCESS,(
null),C:¥Program Files¥Microsoft¥WaterMark.exe
```

図 6 保全情報（f..028mgr.exe PID:3440）

```
01:57:23.295,1,3460,-,PID,3460,C:¥Program
Files¥Microsoft¥WaterMark.exe
01:57:23.295,1,3460,-,PPID,3424,
01:57:23.311,2,3460,kernel32_CreateProcess,PROCESS,(
null),C:¥Program Files¥Microsoft¥WaterMarkmgr.exe
01:57:23.358,3,3460,kernel32_CreateProcess,PROCESS,(
null),C:¥Windows¥system32¥svchost.exe
01:57:26.386,4,3460,kernel32_CreateProcess,PROCESS,(
null),C:¥Windows¥system32¥svchost.exe
```

図 7 保全情報（WaterMark.exe PID:3460）

```
01:57:23.295,1,3468,-,PID,3468,C:¥Program
Files¥Microsoft¥WaterMark.exe
01:57:23.295,1,3468,-,PPID,3440,
01:57:23.311,2,3468,kernel32_CreateProcess,PROCESS,(
null),C:¥Program Files¥Microsoft¥WaterMarkmgr.exe
01:57:23.342,3,3468,kernel32_CreateProcess,PROCESS,(
null),C:¥Windows¥system32¥svchost.exe
01:57:26.370,4,3468,kernel32_CreateProcess,PROCESS,(
null),C:¥Windows¥system32¥svchost.exe
```

図 8 保全情報（WaterMark.exe PID:3468）

```
01:57:23.373,1,3512,-,PID,3512,C:¥Windows¥system32¥s
vchost.exe
01:57:23.389,1,3512,-,PPID,3468,C:¥Program
Files¥Microsoft¥WaterMark.exe
01:57:23.451,2,3512,ws2_32_gethostbyname,NAME,goog
le.com,173.194.126.168,…  ,173.194.126.163
01:58:04.604,4,3512,ws2_32_gethostbyname,NAME,
rtery***tutnrsbberve.com,195.**.26.232
01:58:12.188,6,3512,ws2_32_send,SEND,172.**.143.160:
49163,195. **.26.232:443,77,""
01:58:32.677,7,3512,ws2_32_gethostbyname,NAME,
erwbt***thetcwerc.com,195. **.26.232
…
01:59:03.053,10,3512,ws2_32_gethostbyname,NAME,
rvbwt***twjeitv.com,166. **.62.91
01:59:05.362,11,3512,ws2_32_send,SEND,172.**.143.16
0:49167,166. **.62.91:443,6,""
01:59:05.362,12,3512,ws2_32_send,SEND,172.**.143.16
0:49167,166. **.62.91:443,1,""
02:04:12.859,27,3512,ws2_32_gethostbyname,NAME,
erwbt***thetcwerc.com,195. **.26.232
02:04:14.294,28,3512,ws2_32_send,SEND,172.**.143.16
0:49178,195. **.26.232:443,6,""
02:04:14.294,29,3512,ws2_32_send,SEND,172.**.143.16
0:49178,195. **.26.232:443,77,""
02:04:44.496,31,3512,ws2_32_gethostbyname,NAME,
rvbwt***twjeitv.com,166. **.62.91
…
```

図 9 保全情報（svchost.exe PID:3512）

図 5～図 9 より，当該検体そのもののプロセス（PID:3424）は何ら通信を行わず，検体 1 から CreateProcess API により WaterMark.exe(PID: 3468)プロセスが起動され，さらに当該プロセスから同様に CreateProcess API により起動された svchost.exe(PID: 3512)のみが外部に通信していることが分かる．

図 9 の通信手続き内容を確認すると，DNS 通信や SEND による通信を確認することができ，ネットワーク監視製品により DNS や通信先 IP アドレス，ポート番号等によってアラートが発生した際，保全情報と即座に突合することができ，かつプロセスの依存関係により起点となったマルウェア，新たに起動されたプロセスや作成されたファイル等を，容易に把握することが可能であると言える．さらに，DNS に着目すると“google.com”を正引き後，“rtery***tutnrsbberve.com”，“erwbt***thetcwerc.com”，“rvbwt***twjeitv.com”といった正引きを何度も行い，その都度 443 ポートに対しデータの送信処理を行っている．DNS 等の情報も保全されているため，ネットワーク監視製品が DNS 情報にてアラートを発行した際も，効果を発揮すると言える．

また，ホストベース監視製品がファイルや起動プロセスにてアラートを挙げた場合，こちらも保全情報と突合が可能であり，関連プロセスや起点となったマルウェアの特定，さらには当該不正プロセスの通信手続き情報より DNS や通信先 IP アドレス，ポート番号等をネットワーク監視製品にフィードバックすることも可能となる．なお，Host OS 側ならびに Guest OS 側でパケットキャプチャを行い，プロセスの通信手続きにおける保全情報が正しいことを目視にて確認している．

4.3.2. 検体 2

検体 2 である WS.Reputation.1[9]（Symantec 社名）は我々が入手した実際の標的型攻撃で利用された検体である．提案ツールにて保全したプロセスの通信手続き情報の抜粋を図 11，図 12 に示す．また，保全情報を利用したプロセスの相関関係を図 10 に示す．

```
検体 2_PID:1100
iexplore.exe (PID:4088)
 └ iexplore.exe (PID:1976)
```

図 10 プロセスの相関関係

```
18:11:02.175,1,1100,-,PID,1100,C:¥malware¥検体 2.exe
…
18:11:02.206,2,1100,ws2_32_gethostbyname,NAME,WIN
-LG5MP2BAC3K,172.**.143.160
```

図 11 保全情報（検体 2.exe PID:1100）

```
18:11:02.331,1,1976,-,PID,1976,C:¥Program
Files¥Internet Explorer¥iexplore.exe
18:11:02.346,1,1976,-,PPID,4088,C:¥Program
Files¥Internet Explorer¥iexplore.exe
18:11:02.487,8,1976,ws2_32_GetAddrInfoW,NAME,
winter-****s.net
18:11:03.376,10,1976,ws2_32_GetAddrInfoW,NAME,
winter-****s.net,127.0.0.1
18:11:03.376,12,1976,ws2_32_send,SEND,127.0.0.1:6312
9,127.0.0.1:63129,1,""
18:11:03.376,13,1976,ws2_32_recv,RECV,127.0.0.1:6312
9,127.0.0.1:63129,1,""
…
```

図 12 保全情報（iexplore.exe PID:1976）

当該検体（PID:1100）は感染と同時に iexplore.exe（PID:4088）プロセスに子プロセス（PID:1976）を作成する．その後，当該検体は特にネットワークアクセスは行わないことが分かる．

次に図 12 より，作成された子プロセス（PID:1976）は GetAddrInfoW API を利用して winter-****s.net に対して名前解決を行い，ループバックアドレスである 127.0.0.1 が返答されていることが分かる．さらに，当該プロセスは 127.0.0.1 に対し，ws2_32.dll の send および recv API を利用して 63129 ポートに対してデータの送受信を 11 回実施し，その後自動停止する． 4.3.1 節と同様，保全したプロセスの通信手続き情報を利用することで，ネットワークならびにホストベース監視製品のアラート情報と突合できることが分かる．

4.3.3. 検体 3

検体 3 である Trojan.Win32.AntiFW.b [10]（Kaspersky 社名）を実行した際のプロセスの相関関係を図 13 に示す．また，提案ツールにて保全したプロセスの通信手続き情報の一部を図 14，図 15 に示す．

```
検体 3(PID: 3724)
 └C:¥Program..¥Internet Explorer¥iexplore.exe (PID: 2904)
   └C:¥Program..¥Internet Explorer¥iexplore.exe (PID: 708)
 └C:¥Windows¥explorer.exe (PID: 2804)
 └C:¥Windows¥system32¥cmd.exe (PID: 1688)
 └C:¥Users..¥Temp¥{4..48B}¥Addons¥usetup.exe (PID: 2320)
   └C:¥ProgramData¥..¥PC_Booster.exe(PID: 4068)
 └C:¥Users¥..¥¥Temp¥{4..B}¥Addons¥putfu.exe (PID: 3976)
   └C:¥Windows¥system32¥rundll32.exe (PID: 888)
```

図 13 プロセスの相関関係

```
13:22:05.743,1,3724,-,PID,3724,C:¥mal3.exe.exe
13:22:06.009,2,3724,ws2_32_GetAddrInfoW,NAME,c1.se
te***new.info
13:22:06.009,4,3724,ws2_32_GetAddrInfoW,NAME,r1.ho
me***tmy.info
13:22:06.102,8,3724,ws2_32_GetAddrInfoW,NAME,c1.se
te***new.info,54.191.**.216
13:22:06.258,12,3724,ws2_32_send,SEND,172.**.143.16
0:49165,54.191.**.216:80,540,"GET
/?step_id=1&installer_id=1455
…
ware_id=1301837720965410822 0&prod"
13:22:06.305,13,3724,ws2_32_send,SEND,172.**.143.16
0:49166,54.191.**.197:80,192,"POST
/?report_version=5& HTTP/1.1"
13:22:06.305,14,3724,ws2_32_send,SEND,172.**.143.16
0:49166,54.191.**.197:80,1171,"data=QkP…wr"
…
```

図 14 保全情報（検体 3 PID:3724）

```
13:24:40.648,1,3976,-,PID,3976,
C:¥Users¥..¥¥Temp¥{4..B}¥Addons¥putfu.exe
13:24:40.648,1,3976,-,PPID,3724,C:¥malware¥d…f7.exe
13:24:40.851,2,3976,kernel32_CreateProcess,PROCESS,C
:¥Windows¥system32¥rundll32.exe,"C:¥Windows¥system
32¥rundll32.exe"
"c:¥progra~1¥pc_boo~1¥AssistantSvc.dll",service -install
13:24:41.990,3,3976,ws2_32_GetAddrInfoW,NAME,data
downloadscan.info
13:24:42.068,7,3976,ws2_32_GetAddrInfoW,NAME,data
downloadscan.info,162.210.***.21
13:24:42.271,9,3976,ws2_32_send,SEND,172.**.143.160:
49182,162.210.***.21:80,676,"GET /get/?data=45 …
YLiY3t9jfIbh"
13:24:42.708,10,3976,ws2_32_recv,RECV,172.**.143.16
0:49182,162.210.***.21:80,117,"HTTP/1.1 200 OK"
```

図 15 保全情報（putfu.exe PID: 3976）

当該検体は実行時に installer のような GUI が起動され，検証時はあえて install ボタンを押下して実験を行った．PID:3724，2904，708，2320，3976，888 が示す各プロセスが通信していた．プロセス毎の通信手続きを保全しているため，例えば PID:3724 は表示された GUI が通信した際の保全情報であり，PID:2320 はダウンロードされたファイルの起動プロセスの通信手続きが保全されており，通信を切り分けて把握することができ，比較的容易に分析することができる．また，4.3.1 節と同様，保全したプロセスの通信手続き情報を利用することで，ネットワークならびにホストベース監視製品のアラート情報と突合できることが分かる．

5. 考察

本節では，プロセスの通信手続き情報を利用した不

正プロセスの特定方法と，提案ツールでは保全ができないケースについて考察する．

5.1. 保全ログ情報からの不正プロセス特定手法

提案ツールにて保全した情報から，DNS レスポンスに着目した不正プロセスの特定方法について考察する．

4.3.2 節の検体 2 の保全結果より，DNS のレスポンスにループバックアドレスである 127.0.0.1 が確認できる．なお，論文執筆時に nslookup コマンドを利用し，当該ドメインの正引き結果が 127.0.0.1 であることを確認している．特に外部ドメインの名前解決の際に，このようなループバックアドレスが返答さることは考えにくく，以下二つの要因があると言われている．

- 攻撃者が何らかの目的で DNS にループバックアドレスを設定している
- マルウェア等が利用する不正な DNS に対して，プロバイダ等が被害拡大を防ぐためにループバックアドレスを設定している

両方とも，不正なプロセスを特定するための有用な情報であると言える．これより，保全したプロセスの通信手続きを利用すれば，外部ドメインを正引きした際にループバック IP アドレス（127.0.0.0/8）が返答された疑わしいプロセスを容易に特定することができる．さらに，4.3.1 の保全情報より“rtery***tutnrsbberve.com”と“rvbwt***twjeitv.com”等，複数の異なるドメインを正引きした結果，同じ IP アドレス（195.**.26.232）が返答されていることが分かる．これは，DGA（Domain Generation Algorithm）を利用したマルウェアの通信である可能性が高い．これらも，保全情報より容易に特定することができる．他のプロセスの通信と切り分けて通信手続きを保全しているため，このような通信の特徴も比較的容易に把握できることも本手法のメリットである．また，DNS の名前解決の結果は常に変わりうるため，その結果を保全しておくことにも意味がある．特に，悪性ドメインなどは頻繁に IP アドレスが変更されるため，疑わしい通信が行われた際に解決された DNS クエリとそのレスポンスの値，IP アドレスを保全することはとても重要である．以上より，保全したプロセスの通信手続き情報を利用することで，疑わしいプロセスの検知に応用できると言える．

5.2. 提案手法では保全できないケースについて

提案手法ではプロセスの通信手続きを保全できないケースについて，具体例を挙げて考察する．

4.3.2 節の検体 2 における検体実行時のプロセスの相関関係（図 10）からも分かる通り，検体 2（PID 1100）と検体 2 が新たに起動した iexplore.exe (PID:1976)は，厳密に依存関係が分からない．これは，検体 2 を静的解析した結果，提案システムがフックするプロセスの起動やプロセスインジェクションでは無く，別の方法にて既に起動している iexplore.exe (PID:4088)に対して子プロセスを起動したためである．このような場合は，現状の方式では新たに起動されたプロセスを自動的に追うことができないため，起動されたプロセスに対して，後からフック処理を設け通信手続き情報を保全する形となる．ただし，フック処理を設けるまでの間の通信手続きは保全できない．

6. まとめ

本稿では，プロセスの通信手続きを保全することでプロセスと通信を結び付け，ネットワークおよびホストベース監視製品のアラート情報やログ情報と突合することによる効果的なフォレンジック手法を提案した．また，実際のマルウェアを利用して検証し，提案手法の有効性を示した．また今回は，プロセスの通信手続きの一つである DNS クエリに着目し，提案手法の有効性やプロセスの保全情報を利用した不正なプロセスの特定方法を考察した．

今後の課題としては，提案ツールでは保全できなかったマルウェア検体に対する精度の向上，ならびに実際のセキュリティ製品を利用して様々なアラートを発行させ，提案手法の更なる有効性を検証することである．

文　献

[1] 山本 匠，河内 清人，桜井 鐘治,“不審プロセス特定手法の提案”,Computer Security Symposium 2013.

[2] 山本 匠，河内 清人，桜井 鐘治,“不審プロセス特定手法の実装及び評価”，情報処理学会 CSEC,2014-CSEC-64(33), 1-8, 2014-02-27.

[3] 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”，DICOMO2014.

[4] Wireshark，https://www.wireshark.org/.

[5] Windows Sysinternals，Process Monitor http://technet.microsoft.com/ja-jp/sysinternals/bb896645.aspx.

[6] Process Hacker,http://processhacker.sourceforge.net

[7] RFC 1035 － （DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION） https://www.ietf.org/rfc/rfc1035.txt.

[8] Symantec テクニカルレポート :W32.Ramnit.B!inf http://www.symantec.com/ja/jp/security_response/writeup.jsp?docid=2010-111108-4146-99.

[9] Symantec テクニカルレポート : WS.Reputation.1 http://www.symantec.com/ja/jp/security_response/writeup.jsp?docid=2010-051308-1854-99.

[10] Kaspersky Lab: :W32.Sality.AE http://www.thaikaspersky.com/virus_statistics/february2014.php.