情報処理学会論文誌 Vol.57 No.9 1944–1953 (Sep. 2016)

プロセス情報と関連づけた通信情報保全手法の提案

三村 聡志[1] 佐々木 良一[1,a)]

受付日 2015年12月2日, 採録日 2016年6月2日

概要：サイバー攻撃における原因調査では様々な情報を照らし合わせ，当時の状況を推測してタイムラインを作成し，原因の判定を行うことが必要となる．だが，情報には後から取得可能な情報以外に，揮発性情報と呼ばれる時間の経過にともなって情報の取得が困難になる情報が存在し，対象コンピュータの操作や電源断等によって簡単に消えてしまうという問題点がある．この問題に対処するために，著者らはプロセスの立ち上げや終了，そしてそのプロセスが接続を確立した接続先の情報を，安全にかつシステムにあまり負荷をかけずに記録する方式を提案する．さらに，著者らは，上記の手法を実現する等のために開発したOnmitsuと名付けたドライバプログラムについても報告する．このプログラムを実際の問題に適用することにより，このプログラムが目的を達成することが確認できた．本論文では，提案手法，開発したプログラム，適用結果，ならびにパフォーマンスに関する評価結果を報告する．

キーワード：WFP，Windows，Kernel Driver，Forensics，Logging

Proposal and Evaluation of the Preservation Method of the Network Packets Associated with Process Information

Satoshi Mimura[1] Ryoichi Sasaki[1,a)]

Received: December 2, 2015, Accepted: June 2, 2016

Abstract: For the cause investigation of cyber attack, the cause should be identified by using the timeline created from various information for estimating the status of the cyber attack moment. However, there is a problem that some information called “Volatile Information” will be lost easily by some operation to the computer or computer shutdown. To cope with the problem, the authors will propose a dedicated method for storing packet logs based on the communication, startup and closing log data of the process using Windows functions. In addition, we will report on a newly developed driver program called Onmitsu that can be used to implement the functions included in the proposed method. Based on the results of the application evaluation, it was confirmed that the program could effectively achieve the desired objectives. In this paper, the proposed method, the developed program, applied results, and the evaluation performance results are described.

Keywords: WFP, Windows, Kernel Driver, Forensics, Logging

1. はじめに

特定企業や組織に対し，執拗に機密情報を盗み出そうとする標的型攻撃が世界的に増加している [1]．攻撃では，標的とする組織内のコンピュータに対し外部と通信する機能を持ったマルウェアをメールやUSBメモリ等を介して感染させ，そのうえで，より機能を持ったマルウェアの導入やコンピュータの遠隔操作等を通して機密情報を取得することが行われる．

このような攻撃への対策として，侵入検知システム（IDS）やファイアウォール，アンチマルウェアソフトウェア等の対策製品が登場している．またこれら製品の挙動を統括的に監視して攻撃の検知を実施するSIEMをはじめとする製品や，有事の際の事実関係の把握や法的証拠として通信内容を利用できるようにするために，通信内容を記録する通信保全機器等も登場している．

[1] 東京電機大学
Tokyo Denki University, Adachi, Tokyo 120–8511, Japan
[a)] sasaki@im.dendai.ac.jp

これらの機器により保全された情報は，インシデントが発生した際においては原因の究明や対策方法の立案に必要な情報として利用される．

だがこれらの機器で記録される情報はそのままの形で突合することが難しい場合が多く，一例としてはネットワーク通信とその通信を発生させたプロセスの突合があげられる．

この突合を実施する場合，まず通信が発生したコンピュータを特定し，その後そのコンピュータ内におけるログやnetstat をはじめとする OS のコマンドを実行することで現時点においてどのプロセスがどのポートを開放しているかというような追加情報を収集し，最終的にそれらの情報を総合することにより，当該通信を発生させたプロセスを推測することとなる．

だがこの手法では，どのプロセスが当該通信を実施したかを断言することや，どのような経緯でそのような通信が実施されたかが不明である．

そこで本論文ではこのような現状に対し，プロセスの起動情報，終了，読み込んだモジュール，そしてプロセスが通信を確立させた際の通信情報について逐次記録する具体的な手法を Windows 環境向けに提案する．

この 4 項目を回収することにより，2012 年に発生した遠隔操作ウィルス事件 [15] で用いられた Backdoor.Rabasheeta [14] のようなマルウェアに感染していた場合においてその通信を発生させたプロセスの特定とそのプロセスを実行させた親を特定することができるため，ブラウザからのウェブサイト閲覧の通信であっても，それがコンピュータの利用者によるものか，マルウェアが発生させたものかを区別することができるようになる．

また，本提案手法を実際に適用した場合における性能評価ならびに提案手法の活用方法についての検討結果も報告する．

2. 関連研究

すでに不審な通信からプロセスを特定する手法はいくつか提案されている．まず，山本ら [6], [7] により提案されている手法では IDS による検知をきっかけとして特定を開始する．特定にはコンピュータ内部で netstat コマンドを実行した出力結果とポート番号を照合した結果を用いる．照合した後，当該プロセスの情報を解析システムに送信し，改竄の有無により不審なプロセスかどうかの検証を行っている．

この手法では，IDS のアラートをトリガとしているが，IDS での検知に失敗した場合において，情報収集が開始されないという欠点が存在している．また，アラートを受信してから調査を実施するために，すでにプロセスの隠蔽がマルウェアによって実行されていた場合に検出できない可能性があることに加え，当該プロセスがどのような経緯で起動したかが不明となっている点があげられる．

表 1 他の手法と本論文の方式の CPU にかかる負荷の違い
Table 1 Difference of the load to CPU between proposed method and the others.

	なし	本手法	Process Monitor	Network Analyzer
CPU 利用率（5 回平均 %）	7.642	8.789	17.188	18.164

筆者らが提案する手法ではカーネルドライバという形でシステム内に導入し，起動時からプロセスとそのプロセスが発信する通信に関しての動作ログをとり続ける手法をとっている．そのため，山本らの手法と比較すると取得開始のトリガがない分ログサイズは大きくなる傾向にあるが，マルウェアによる改竄が行われにくいカーネル空間で動作するため，マルウェアによるプロセス情報の隠蔽処理を回避できる可能性が高いといえる．

また同様にカーネルドライバという形でログの取得を実行する手法として，神園ら [16] が提案する手法も存在している．この手法では本手法と同じくカーネルドライバとして常時稼働する手法をとるが，通信の回収手法として Windows API をフックすることで情報の回収を実現している．

Windows API のフックによる手法では，フックを実現させるためにプロセスの処理内容に対し外部から書き換えを実行する必要がある一方で，プロセスによる API の呼び出しをすぐに把握できることに加え，呼び出し時におけるパラメータを確認することもできるため，よりプロセスの処理に踏み込んだ形で動作を監視することが可能である．

だが，プロセスの処理内容に一定の書き換えを実行することは，プロセスの状態が製造元が意図した状態とは異なる状態になることでもあり，一般業務で使用されるコンピュータに対して手法を適用した場合において，非適用状態と適用状態のどちらにおいてもプロセスが正常に稼働させることが難しくなるリスクが存在するといえる．

その他，常時通信やプロセス情報を記録し続けるものとして Process Monitor [17] や Message Analyzer [22] が存在する．Process Monitor はプロセスに関する詳細な情報が取得でき，また Message Analyzer においてはプロセスとその通信情報に関して詳細な情報が取得できる．だがこれらのソフトウェアは常時稼働させ，ログを記録し続けるという点においては CPU 利用率が**表 1** のように高いリソースを必要とすることに加え，Process Monitor や Message Analyzer はログをファイルにつねに書き出し続けるソフトウェアではなく，メモリ上に記録していくため任意のタイミングで手動で保存操作を実行する必要があるため，無人操作でのログの常時取得目的としては使いにくいという問題点がある．

ほかにも本手法と似たような機能を提供するものとしてSysmon [18] と呼ばれるツールが存在する．このツールで出力されるログはWindowsのイベントログの形式をとっており，本手法で用いているCSV形式と比較すると汎用のテキストエディタやテキスト処理ソフトで直接扱えないという問題点が存在する．

関連研究およびツールと提案手法の明確な差違は，有事の際に有用なログをつねに記録し続けることができる一方で，導入された環境において非導入状態と変わらない可用性を追求していることであり，可能な限りシステムへの負荷の低減と，他のプロセスへの干渉をしないようにすることでこれを実現している点が他の手法と大きく異なっている．またSysmonより本手法のほうが先行研究となっており，新規性が存在する．すなわち，本研究に関しては2014年7月11日に開催されたDICOMO 2014 [20] において発表しており，製品としても2014年7月にプレスリリースし，同年8月1日より発売している [21]．一方，Sysmonは2014年8月5日（米国時間）に公開 [19] となっており，本方式が早いことがいえる．

3. 提案方式

3.1 提案方式の概要

本手法を適用する想定環境を簡略化した図を**図 1** に示す．

想定環境は企業内のネットワークに複数台のコンピュータが接続されている環境を想定している．

DMZ（非武装地帯）に各種サーバ群が接続されており，また業務上必要な機密情報等が記録されるサーバが内部ネットワークに接続されている．

情報の取得にはMicrosoft Windows VistaもしくはMicrosoft Windows Server 2008以降のオペレーティングシステムに標準搭載されているAPIを使用するため，本手法を適用するコンピュータは先述のバージョン以降のWindowsが実行されていることが必要となる．

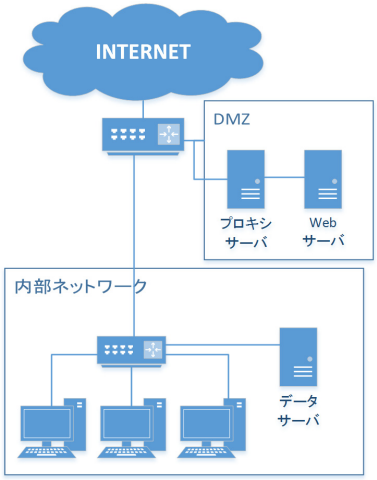

図 1 想定ネットワーク環境
Fig. 1 Object network environment.

本手法では**表 2** に示す項目をCSV形式のログファイルに出力する．CSV形式は，Microsoft Excelをはじめとする多くのソフトウェアで解析可能であり，また導入環境の利用者自身が表2に示す項目以外を取得していないことを確認できることから採用した．

3.2 提案方式の詳細

本手法では表2において示した各挙動を**表 3** に示すAPIを使用して情報を取得する．なお，これらはすべてWindowsにおいて標準で用意されているAPIであり，プロセスに対する個別のフックやコード注入を行わず，OS側が提供する情報を用いる．

そして，これらAPIの組合せを図にしたものが**図 2** である．

まず，プロセス情報の取得に関してはPsSetCreateProcessNotifyRoutineExおよびPsSetLoadImageNotifyRoutine関数を用いる．取得のためには上記2つの関数に対応するコールバック関数をドライバ側で用意し，それらをWindowsに登録する．これにより，プロセスが新規で起動しようとした際にコールバック関数を通してWindowsからアプリケーション起動およびモジュール読み込みの情報

表 2 ログに記録される情報一覧
Table 2 List of information recorded in log.

挙動	記録内容
プロセス起動	起動時刻
	プロセス ID
	要求を行った親プロセス ID
	実行イメージファイルパス
	コマンドライン
プロセス終了	終了時刻
モジュール読み込み	読み込んだ時刻
	プロセス ID
	モジュールイメージパス
ネットワーク通信	通信確立時刻
	プロセス ID
	接続元 IP アドレス
	接続元ポート番号
	接続先 IP アドレス
	接続先ポート番号
	トランスポート層プロトコル ID

表 3 各挙動の動作内容を取得するために用いる API
Table 3 API used to collect behavioral elements.

挙動	取得方法
プロセス起動	PsSetCreateProcessNotifyRoutineEx [4]
プロセス終了	
モジュール読込	PsSetLoadImageNotifyRoutine [5]
接続確立（TCP）	Windows Filtering Platform [3]
送受信（UDP，ICMP）	Windows Filtering Platform [3]

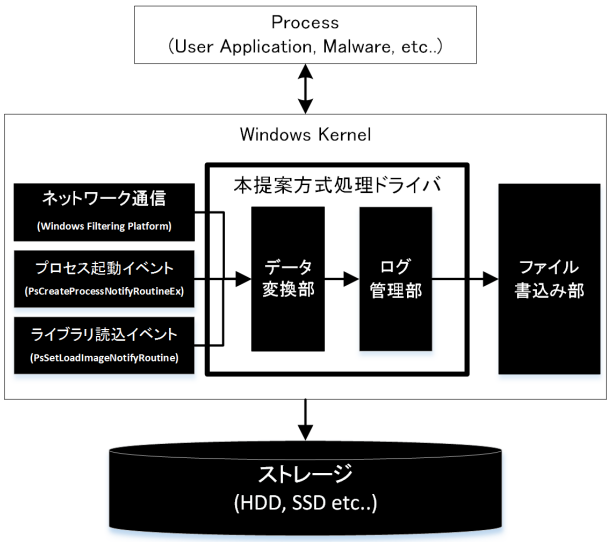

図 2 提案方式の構造
Fig. 2 Structure of proposed method.

が取得できるようになる．これら手法はいくつかの類似論文 [8], [9], [10] において利用されている手法である．

続いて，通信に関しては Windows Filtering Platform を使用して情報の取得を行う．これもプロセス情報の取得と同様対応するコールバック関数を用意しておき，ファイアウォールの通信判断用プログラムとしてそれを登録することで実現する．

具体的には，通信確立時に判定を行うファイアウォールとして動作させるため，FWPM_LAYER_ALE_FLOW_ESTABLISHED というトリガにより呼び出されるように，コールバック関数をシステムに登録する．その後，通信が発生しコールバック関数が呼び出された際は情報を回収すると同時に，FWP_ACTION_CONTINUE というフラグをシステムに返し，システムに導入されているファイアウォールやウィルス対策ソフトに対して通信の可否の判断をゆだねるようにしている．

これら API で収集した情報はカンマ区切りのログ情報に変換し，ログファイルに対し追記を行う．ログファイルはドライバ起動時にファイルをロックし，他のプロセスからの改竄や閲覧が行えないようにして書き込みを行う．

4. 評価

4.1 評価環境

表 4 に評価に使用したソフトウェアを示す．

評価は同一の計算機上に Microsoft Hyper-V を用いて 2 つの仮想計算機を用意して実施した．想定環境として提示した図 1 中に示す環境と評価環境は異なるが，本手法はあくまでも単一のコンピュータ上において発生する動作について記録を行う手法であるため，本章での評価時は，評価可能な最小構成で行った．

また，本提案手法を実現するカーネルモードドライバは

表 4 評価に使用したソフトウェアおよび機材
Table 4 Software and equipment used for evaluation.

種別	製品名
仮想マシン	Microsoft Hyper-V
	Oracle VirtualBox 4.0.30
サーバ	Microsoft Windows 8.1 Pro（x86_64）
	Ubuntu 14.04
クライアント	Microsoft Windows 7（x86_64）
回線	Hyper-V 仮想ネットワークアダプタ
	DoCoMo Xi 回線
通信ソフトウェア	ping（OS 標準搭載）
評価用ツール	NetIO 1.3
	Futuremark PCMark 8
評価用マルウェア	TROJ_MIRUEF.LWR（TrendMicro 表記）
	VirusShare_00197.zip
開発用ソフトウェア	Microsoft Visual Studio 2013

Windows Driver Frameworks [2] および Microsoft Visual Studio 2015 を用いて作成し，ドライバ名およびサービス名はともに “Onmitsu” として実験環境に導入した．

作成したドライバは net start および net stop コマンドによりドライバを制御できるように実装し，評価時は，評価用のソフトウェアを起動する直前にドライバ開始し，ログを取得できるようにした．

また，実際のマルウェアにおける動作確認目的として TrendMicro の表記で “TROJ_MIRUEF.LWR” と表記される一般に “Zeus” と呼ばれているマルウェアを用意し，ログの取得内容の確認を行った．

そして性能評価目的として VirusShare [23] より入手可能な VirusShare_00197.zip に含まれている検体よりランダム選択した 9,000 検体のマルウェアを Cuckoo [24] 上で動作させることで，取得漏れがないかどうかを確認した．

4.2 パフォーマンスの評価結果

まず ping コマンドを用いて正常に記録できているかを確認した．実験時においての www.google.co.jp のアドレスは 173.194.126.233 であった．Windows 標準の ping コマンドにおいて回数を指定しない場合は 4 回送信する仕様となっている．

発信元プロセスに関しては ping コマンドの内部で使用している IcmpSendEcho2Ex 関数の仕様により，System（PID:4）が実行しているように記録される．

また名前解決のクエリに関しては，Windows のデフォルト状態では DNS Client サービス（Dnscache）がまとめて代理として問合せを行うため，本記録からは除外している．

結果は次に示すとおり 173.194.126.233 への通信を 4 回分正常に記録できていることが確認できる．結果を**表 5** および**表 6** に示す．

続いて次の**図 3** に示す環境を用い，取得時にシステムに

表 5 プロセス起動時のログ
Table 5 Log collected at the time of starting the process.

TYPE	PID	PARENT	CMDLINE
LAUNCH	4348	6256	ping www.google.co.jp
MODLOAD	4348		¥Device¥HarddiskVolume2 ¥Windows ¥System32 ¥PING.EXE

表 6 通信記録に該当するプロセス通信記録の抜粋
Table 6 Abstract of process communication record related to communication record.

TYPE	PID	SRC	I_TYPE	DST
NETWORKV4	4	192.168.0.39	8	173.194.126.223
NETWORKV4	4	192.168.0.39	8	173.194.126.223
NETWORKV4	4	192.168.0.39	8	173.194.126.223
NETWORKV4	4	192.168.0.39	8	173.194.126.223

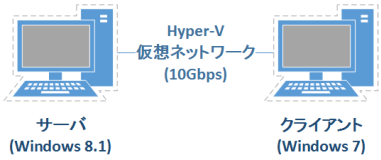

図 3 システム負荷計測時のネットワーク構成
Fig. 3 Network structure at the time of the system load measurement.

表 7 TCP を用いた受信方向の速度評価（単位：Kbps）
Table 7 Evaluation of receiving speed while using TCP (Kbps).

転送量	非適用時（A）	適用時（B）	B/A
1 KB	3,421,306	3,317,350	97%
2 KB	4,501,995	4,049,715	90%
4 KB	5,462,999	5,131,468	93%
8 KB	6,041,600	4,940,513	82%
16 KB	6,761,349	5,961,482	88%
32 KB	7,669,186	7,273,267	95%
合計	33,858,435	30,493,795	90%

かける負荷を確認した．

ネットワーク通信に関わる負荷として TCP，UDP それぞれの送受信方向の速度を NetIO を用いて計測を行った．サーバマシンとクライアントマシンは Hyper-V が提供する仮想ネットワークアダプタで相互に接続し，速度は 10 Gbps として接続を行った．計測は 30 回実行し数値はその平均値とした．

まず，TCP 通信を用いて性能評価を実施した結果を**表 7** および**表 8** に示す．

次に，UDP 通信を用いて性能評価を実施した結果を**表 9** および**表 10** に示す．

上記結果より，提案手法を適用した場合には最大 18%の速度低下がみられることが判明した．ただし，一般に普及しているネットワーク回線が 1 Gbps の回線であることを

表 8 TCP を用いた送信方向の速度評価（単位：Kbps）
Table 8 Evaluation of sending speed while using TCP (Kbps).

転送量	非適用時（A）	適用時（B）	B/A
1 KB	3,643,064	3,464,232	95%
2 KB	5,066,752	4,625,285	91%
4 KB	6,180,536	5,891,194	95%
8 KB	6,585,139	5,807,226	88%
16 KB	7,257,866	6,383,042	87%
32 KB	7,407,861	7,256,719	97%
合計	36,141,218	33,427,698	92%

表 9 UDP を用いた受信方向の速度評価（単位：Kbps）
Table 9 Evaluation of receiving speed while using UDP (Kbps).

転送量	非適用時（A）	適用時（B）	B/A
1 KB	1,961,574	1,758,494	90%
2 KB	2,062,909	1,970,995	96%
4 KB	3,622,256	3,514,531	97%
8 KB	1,633,730	1,482,424	91%
16 KB	2,516,664	2,489,958	98%
32 KB	3,211,427	3,051,028	95%
合計	15,008,560	14,267,430	95%

表 10 UDP を用いた送信方向の速度評価（単位：Kbps）
Table 10 Evaluation of sending speed while using UDP (Kbps).

転送量	非適用時（A）	適用時（B）	B/A
1 KB	2,627,256	2,147,205	82%
2 KB	2,192,261	2,113,863	96%
4 KB	3,615,457	3,614,146	99%
8 KB	1,758,904	1,740,390	99%
16 KB	2,643,558	2,604,810	99%
32 KB	3,583,180	3,239,526	90%
合計	16,420,616	15,459,940	94%

考えると，どの計測結果も 1 Gbps は適用時および非適用時双方において超えており，通常利用においては問題にはならないと考えられる．

次にシステムにかかる負荷について Futuremark PCMark 8（Futuremark，Espoo，Finland）というソフトウェアよる Office Benchmark を用いて評価を実施した．この評価手法はインターネットブラウザによるウェブサイト閲覧および文書作成ソフトウェアによる文書入力と同じ操作を機械的に実行し，その際のコンピュータのパフォーマンスを数値として評価するものであり，一般のオフィス環境におけるコンピュータ作業で行われるパフォーマンスを計測する手法として使われる手法である [25]．この評価手法で本手法を評価した結果を**表 11** に示す．

上記の結果から本手法のドライバ導入によるシステム負荷は比として 0.01 の差であり，大きな差はないといえる．

表 11 Futuremark PCMark 8 Office Benchmark の結果
Table 11 Result of experiment using Futuremark PCMark 8 and Office Benchmark.

条件	スコア	比
非適用時	4319	1.00
適用時	4264	0.99

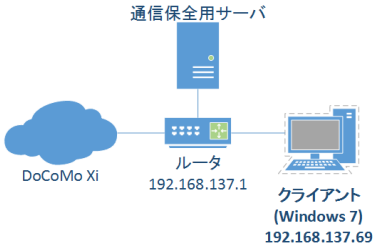

図 4 Zeus マルウェアを用いた評価試験時のネットワーク構成
Fig. 4 Network structure when Zeus malware was used for evaluation experiment.

4.3 Zeus マルウェアを用いた評価

次に，実際のマルウェアを仮想環境上において実行し，ログの取得可能性や取得した情報の評価，記録されるログファイルのサイズについて評価を行った．なおこのマルウェアは事前に C&C サーバに接続不可能なことを確認している．

まずはじめに，Zeus と呼ばれるマルウェアについて実行させたうえでログを収集し，ログが正常に収集されるかを確認した．検証環境を**図 4** に示す．

この実験では表 4 に示した環境のうちクライアント環境のみを用い，本手法のドライバを起動させた後マルウェアを実行してログを取得することで評価を行った．評価はマルウェアを起動した後，10 分間稼働させることで行った．

図中における通信保全用サーバとは，通信のミラーポートより通信内容を取得しそのまま記録するサーバのことを指している．

まず，本手法で取得したログの内容を文末の**表 A·1** に示す．当該時刻の範囲のログはすべてで 629 行あるが，マルウェアの挙動とは関係のない他プロセスの処理や通信が含まれているため，マルウェアに関係ない箇所を省略し当該箇所に灰色の行を挿入した．このログからマルウェアである invoice_928649039284232_9482934d88.pdf.exe を起動した後，Temp フォルダに作成されたマルウェアを起動していることが読み取れる．また Internet Explorer が起動し何らかの通信を行い，その後にマルウェアが通信を行っていることも読み取れる．

次に，通信保全用サーバで記録した通信内容との照合を行った．行われた通信の内容を Wireshark で表示したものを**図 5** に，またこの通信に関連する部分を表 A·1 より抜粋した表を**表 12** および**表 13** に示す．

上記より図 5 において示されている通信のうち，送信元

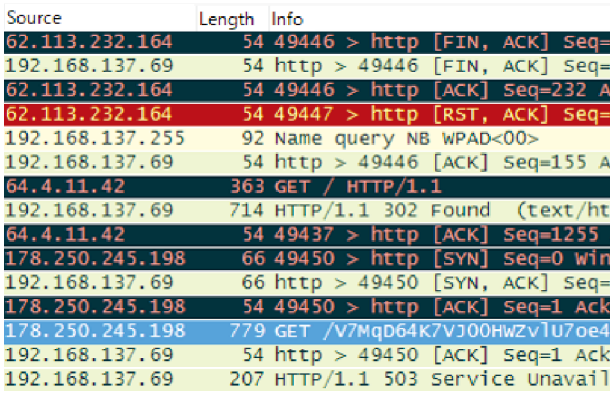

図 5 wireshark での通信表示（抜粋）
Fig. 5 Extracted display of communication measured using Wireshark.

表 12 通信記録に該当するプロセス起動情報の抜粋
Table 12 Abstract of process start information related to communication record.

PID	PARENT	CMDLINE
1832	1848	C:¥Users¥TESTUSER¥Desktop ¥SHARE ¥invoice_928649039284232 _9482934d88.pdf.exe
2068	1832	C:¥Users¥TESTUSER¥AppData ¥local¥Temp ¥zdttuqbg.exe
1896	2068	C:¥Users¥TESTUSER¥AppData ¥local¥Temp ¥zdttuqbg.exe
2716	752	C:¥Program Files¥Internet Explorer ¥iexplore.exe -Embedding

表 13 通信記録に該当する通信ログの抜粋
Table 13 Abstract of communication log related to the communication record.

PID	SRCPORT	DSTIP	DSTPORT
2716	49446	62.113.232.164	80
2716	49447	62.113.232.164	80
1896	49450	178.250.245.198	80

ポートが 49446 もしくは 49447 である通信は表 12 および表 13 より Internet Explorer による通信であることが読み取れる．

また同様に，送信元ポートが 49450 である通信は zdttuqbg.exe による通信であり，このプロセスは表 12 の PARENT 情報を元にさかのぼることで，マルウェアに関するプロセスが発した通信であることも読み取れる．

4.4 Cuckoo サンドボックスを用いた評価

最後に，Cuckoo サンドボックス [24] を用いて Onmitsu におけるログの取得漏れの有無を確認した．マルウェアは VirusShare [23] で入手できる検体のうち，VirusShare_00224.zip に含まれている 9,000 検体のマルウェアを使用した．このファイルは 2016 年 3 月 11 日から

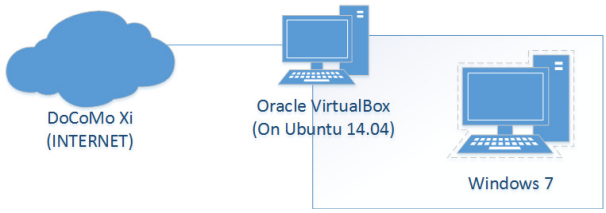

図 6 Cuckoo サンドボックスを用いた評価
Fig. 6 Evaluation using sandbox named Cuckoo.

表 14 Cuckoo のログと Onmitsu のプロセス起動ログ一致数
Table 14 Agreement number on process start in the log from Cuckoo and Onmitsu.

総ログファイル数	8,289
ログ破損数（全検体数 – 総ログファイル数）	711
総レコード数	8,572
双方に存在するレコード	8,572

表 15 Cuckoo のログと Onmitsu のネットワーク通信ログ一致数
Table 15 Agreement number on network communication in the log from Cuckoo and Onmitsu.

総ログファイル数	8,289
ログ破損数（全検体数 – 総ログファイル数）	711
総レコード数	37,477
双方に存在するレコード	28,934
Cuckoo のみに存在するレコード	8,543
接続不可能なホストのレコード	8,543

同年 3 月 19 日までの期間のうち VirusShare に送信されたファイル群となっている．

この評価では Ubuntu 14.04 環境上に Oracle VirtualBox をインストールし，さらにその上に Onmitsu をインストール済みの Windows 7 の仮想マシンを用意し，Cuckoo の解析環境とした．

検証環境を**図 6** に示す．

評価は Cuckoo サンドボックスの出力のうち processtree と network 欄の出力結果を用い，Onmitsu の出力結果上にどれだけ Cuckoo サンドボックスと同じ出力結果が含まれていたかによって評価した．

まず，プロセス起動に関しての結果を**表 14** に示す．

続いてネットワーク通信に関しての結果を**表 15** に示す．

上記より，表 14 に示されているとおり，プロセス起動に関してはすべて記録されていることが分かる．

また，表 15 において Cuckoo のみに存在するレコードが 8,543 件存在した．これは Onmitsu が接続を確立したものについてのみログを記録するのに対し，Cuckoo では接続試行をしたホストに関しても記録することによる仕様上の違いがある．現に，Cuckoo のみに存在するレコードに関してはすべて到達不可能であることを確認した．

以上より，Onmitsu においてはマルウェアの挙動に関しての情報を正常に取得できていると判断できる．

5. 明らかになった課題と対応策

実験を実施する中で本手法に関するいくつかの課題点を見つけることができた．次に課題点を示す．

5.1 ログの肥大化

Onmitsu は機能を有効にしている限りログファイルを出力し続けるため，コンピュータの記録媒体に対して一定の容量が必要となる．本論文執筆時にエディタ操作とブラウザによるウェブサイト閲覧を行いながら Onmitsu を用いて動作を記録したところ，平均 1 時間あたり 3.46 MB であり，1 日 8 時間稼働させると計算すれば 30 MB ほどに肥大化することが予想される．

また本手法ではプロセスの起動回数や通信の実行回数に比例してログのファイルサイズは増大するため，利用状況によっては多くのディスク容量が必要になる可能性が考えられる．

そこで，日が変わるごとに自動的に前日までのログを zip 形式で圧縮を行うことにより，ディスクへの圧迫を軽減する方法を現在検証している．

先述の実験において取得された 10.3 MB のログファイルに対し zip 圧縮を実行したところ 738 KB まで圧縮されることが分かっており，ディスクへの圧迫を軽減できると考えている．

5.2 接続試行動作の記録

本手法ではネットワーク通信のログに関しては，接続が確立した時点でその情報を記録する設計になっている．そのため，4.4 節のネットワーク通信の記録のように，接続試行をする動作がいっさい記録されない状態になっている．

今後接続試行の記録が必要かどうかをログのファイルサイズやパフォーマンス面から検討を進めていく．

6. 今後の展望

以上により，特定のコンピュータ内から発せられた不審な通信から，そのコンピュータ内における通信を実行したプロセスを特定することが可能である見通しを得た．今後，次のような項目を実施していきたいと考えている．

(1) 筆者が開発したプログラムについては，すでに製品化されており [11]，実適用を通じてさらに有益で使いやすいものを目指す．
(2) 開発したプログラムを他のシステムと連携させることを検討している．筆者らは現在ネットワークフォレンジックのための知的ガイドシステム LIFT（Live Intelligent Network Forensics Toolkit）の研究開発を進めており [12], [13]，このシステムの要素の 1 つとして組み込みを検討している．
(3) 各プロセスの DNS 問合せに関し，Windows の DNS

Client サービスに対して情報取得を試みることにより，個別にトレースができるようにする．

7. おわりに

本論文では不審な通信の原因を特定する手法としてプロセス情報とそのプロセスが発した通信に関するログを記録する手法を提案するとともに，その手法を実現するプログラムを実際に開発して実験を行うことにより，基本的有効性を確認することができた．

今後は 5 章で述べた課題の解決策の有効性を検証するとともに，6 章で述べた各活用手法の具体化と評価を実施していく予定である．また，その他の揮発性情報の保全手法に関しても調査研究を続けていきたいと考えている．

参考文献

[1] Kaspersky：Advanced Persistent Threat（APT）攻撃：今までにない高度なマルウェア，入手先 ⟨http://www.kaspersky.co.jp/downloads/pdf/advanced-persistent-threats-not-your-average-malware_kaspersky-endpoint-control-white-paper_jp.pdf⟩（参照 2014-11-18）．

[2] MSDN: Windows Driver Frameworks, available from ⟨http://msdn.microsoft.com/en-us/library/windows/hardware/ff557565(v=vs.85).aspx⟩ (accessed 2014-11-18).

[3] MSDN: Windows Filtering Platform, available from ⟨http://msdn.microsoft.com/en-us/library/windows/desktop/aa366510(v=vs.85).aspx⟩ (accessed 2014-11-18).

[4] MSDN: PsSetCreateProcessNotifyRoutineEx routine, available from ⟨http://msdn.microsoft.com/en-us/library/windows/hardware/ff559953(v=vs.85).aspx⟩ (accessed 2014-11-18).

[5] MSDN: PsSetLoadImageNotifyRoutine routine, available from ⟨http://msdn.microsoft.com/en-us/library/windows/hardware/ff559957(v=vs.85).aspx⟩ (accessed 2014-11-18).

[6] 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).

[7] 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，pp.1–8 (2014).

[8] Tirli, H., Pektas, A., Falcone, Y. and Erdogan, N.: Virmon: A Virtualization-Based Automated Dynamic Malware Analysis System, *INTERNATIONAL INFORMATION SECURITY & CRYPTOLOGY CONFERENCE*, pp.59–64 (2013).

[9] Liu, Z. and Chen, P.: Improved Method of packet Filtering, *International Symposium on Web Information Systems and Applications*, pp.294–296 (2009).

[10] Park, S.: Malware Expert: Execution Tracking, *Cybercrime and Trustworthy Computing Workshop* (2012).

[11] dit：CAP Logger マルウェアの特定と社内感染の追跡を実現，入手先 ⟨http://www.dit.co.jp/products/caplogger/⟩（参照 2014-11-18）．

[12] 佐々木良一，上原哲太郎，松本 隆：標的型攻撃に対するネットワークフォレンジック対策の現状と今後の展望，*Computer Security Symposium* (2013).

[13] 比留間裕幸，橋本一紀，佐々木良一，上原哲太郎，佳山こうせつ，松本 隆，柿崎淑郎，八槇博史：標的型攻撃に対する知的ネットワークフォレンジックシステム LIFT の開発（その 1）—予兆検知と対策指示方法の提案，*DICOMO* (2015).

[14] Symantec: Backdoor.Rabasheeta, available from ⟨http://www.symantec.com/ja/jp/security_response/writeup.jsp?docid=2012-101004-0445-99&tabid=2⟩ (accessed 2015-11-26).

[15] Yahoo：遠隔操作ウィルス事件，入手先 ⟨http://news.yahoo.co.jp/list/?t=remote_control_virus⟩（参照 2015-11-26）．

[16] 神薗雅紀，遠峰隆史，津田 侑：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，Computer Security Symposium (2014).

[17] Windows Sysinternals: Process Monitor, available from ⟨http://technet.microsoft.com/ja-jp/sysinternals/bb896645.aspx⟩ (accessed 2015-11-26).

[18] Windows Sysinternals: Sysmon, available from ⟨http://technet.microsoft.com/ja-jp/sysinternals/dn798348⟩ (accessed 2016-03-02).

[19] 新しい仲間「Sysmon」はトラブルシューティングの必携ツールになりそうな予感，入手先 ⟨http://www.atmarkit.co.jp/ait/articles/1409/02/news011.html⟩（参照 2016-06-05）．

[20] 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，*DICOMO* (2014).

[21] dit：マルウェアの特定と社内感染の追跡を実現「CAPLogger」の販売開始，入手先 ⟨http://www.dit.co.jp/news/news2014/2014_0710.html⟩（参照 2016-06-07）．

[22] Technet: MessageAnalyzer, available from ⟨http://blogs.technet.com/b/messageanalyzer/⟩ (accessed 2015-11-26).

[23] VirusShare, available from ⟨https://virusshare.com/⟩ (accessed 2015-11-26).

[24] Cuckoo Sandbox, available from ⟨http://www.cuckoosandbox.org/⟩ (accessed 2015-11-30).

[25] Jiao, Y. and Wang, W.: Design and Implementation of Load Balancing of Distributed-system-based Web Server, *ISECS, 2010, Electronic Commerce and Security, International Symposium 2010*, pp.337–342, DOI:10.1109/ISECS.2010.81 (2010), available from ⟨http://www.computer.org/csdl/proceedings/isecs/2010/4219/00/4219a337-abs.html⟩ (accessed 2016-03-20).

付　　録

A.1 本手法によるマルウェア動作時の取得ログ

表 A·1 本手法で取得したマルウェア動作ログ

Table A·1 Log using the proposed method at the time of malware operation.

DATE	TYPE	PID	PARENT	CMDLINE	SRCIP	SRCPORT	DSTIP	DSTPORT
2014-02:42:09:396	PROCESS_LAUNCH	1832	1848	C:¥Users¥TESTUSER¥Desktop¥SHARE ¥invoice_928649039284232_9482934d88.pdf.exe				
2014-02:42:09:403	PROCESS_MODLOAD	1832		¥SystemRoot¥System32¥ntdll.dll				
2014-02:42:09:403	PROCESS_MODLOAD	1832		¥SystemRoot¥SysWOW64¥ntdll.dll				
2014-02:42:09:507	PROCESS_LAUNCH	2068	1832	C:¥Users¥TESTUSER¥AppData¥local¥Temp ¥zdttuqbg.exe				
2014-02:42:09:514	PROCESS_MODLOAD	2068		¥SystemRoot¥System32¥ntdll.dll				
2014-02:42:15:15	PROCESS_LAUNCH	1896	2068	C:¥Users¥TESTUSER¥AppData¥local¥Temp ¥zdttuqbg.exe				
2014-02:42:15:21	PROCESS_QUIT	2068						
2014-02:42:15:21	PROCESS_QUIT	1832						
2014-02:42:16:388	PROCESS_LAUNCH	2716	752	C:¥Program Files¥Internet Explorer ¥iexplore.exe -Embedding				
2014-02:42:55:538	NETWORKV4	2716			192.168.137.69	49446	62.113.232.164	80
2014-02:42:55:540	NETWORKV4	2716			192.168.137.69	49447	62.113.232.164	80
2014-02:42:55:938	PROCESS_QUIT	2716						
2014-02:43:12:317	NETWORKV4	1896			192.168.137.69	49450	178.250.245.198	80
2014-02:43:29:551	NETWORKV4	1896			192.168.137.69	49451	144.76.252.231	80

三村 聡志

1992 年生．2014 年東京電機大学未来科学部情報メディア学科卒業．同年より同大学大学院修士課程所属．

佐々木 良一 （正会員）

1971 年 3 月東京大学卒業．同年 4 月日立製作所入社．システム開発研究所にてシステム高信頼化技術，セキュリティ技術，ネットワーク管理システム等の研究開発に従事．2001 年 4 月より東京電機大学教授，工学博士（東京大学）．1998 年電気学会著作賞受賞．2002 年情報処理学会論文賞受賞．2007 年総務大臣表彰等．著書に，『IT リスクの考え方』（岩波新書，2008 年）等．日本セキュリティ・マネジメント学会会長，内閣官房サイバーセキュリティ補佐官．本会フェロー．