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

三村 聡志[†1,a)] 佐々木 良一[†1]

概要：標的型攻撃などの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原因の判定を行う事が必要となる．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伴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などによって簡単に消えてしまう．本論文ではそれら揮発性情報のうち通信とそれを実行するプロセスの情報に着目し，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共に，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，原因となるプロセスを発見できることを確認した．また実験によって明らかになった新たな課題とその対応策も記述している．

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

Proposal of the Cause Estimation Method Using the Network Packets Associated with Process Information

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

1. はじめに

近年，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 メモリなどを介して感染させ，その上で，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など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の導入に加え，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なども登場している．

だがこれらの機器では，攻撃の開始点となったコンピュータやその上で動作していたプロセスの情報，またそれらがどこに対して通信を実施したかという情報に関しては十分に記録されておらず，原因と疑われるコンピュータを特定できても，感染したマルウェアの特定や攻撃に利用されたソフトウェアの特定に時間がかかるために，問題箇所の特定やその後の対策に時間が掛かる問題があった．

本論文ではこのような現状に対し，プロセスにおける起動・終了・モジュール読み込み・通信試行の４項目を，動作を実行したプロセスの情報と関連づけて逐次記録する具体的手法を Windows 環境向けに提案する．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並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

既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．一例として山本ら [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．まずコンピュータ内部にて netstat コマンドを実行した出力結果とポート番号を照合してプロセスを特定し，次に当該プロセスにの情報を解析システムに送信した上で，改竄の有無により不審なプロセスかどうかという検証を行っている．

[†1] 現在，東京電機大学
Presently with Tokyo Denki University
[a)] 14fmi29@ms.dendai.ac.jp

この手法では，IDS のアラートをトリガーとしているが，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，最初の情報として netstat コマンドによる出力を使用しているが，不正通信を検知した時点で当該コンピュータ内ではマルウェアによりプロセスを隠蔽する処理が行われている可能性が高く，当該コマンドを実行したとしても原因プロセスの情報が取得できるとは限らない．そして，アラートを受信してから調査を実施するため，通信を実際に行ったプロセスは判明してもそのプロセスの起動における経緯が分からないという欠点も存在している．

本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通信に関してログをとり続ける手法をとる．そのため，山本らの手法と比較すると取得開始のトリガーがない分ログサイズは大きくなる傾向にあるが，カーネルドライバであることと常に動いているということからマルウェアによるプロセス情報の隠蔽処理を回避することが出来る可能性が高いと言える．また後に問題が発覚した際においても過去の情報をさかのぼって調査を行うことが出来ることが挙げられ，不正通信を検知し調査を開始するまでにプロセスが接続を終了していた場合や，実行ファイルが消去されていた場合に関しても本手法であればログをたどって特定することが可能となる点が異なる．

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．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ある．

内部ネットワークに接続されたコンピュータにおける外部との通信方法は特に定義せず，直接通信を行っても，図に示してある通り DMZ 帯に設置したプロキシサーバを経由しても構わないものとする．

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コンポーネントを使用する為，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 が実行されている事が必要となる．

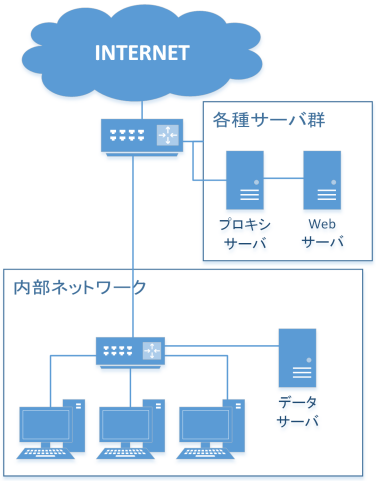

図 1 想定ネットワーク環境

本手法では表 3.1 に示す項目をログへ記録する．また記録されるログは，多くのソフトウェアで解析可能であることや，導入環境の利用者自身が表 3.1 に示す項目以外を取得していない事を確認できることから，暗号化や独自形式による保存はせずに純粋なテキストにより表現される CSV 形式を用いて記録するようにした．

表 1 ログに記録される情報一覧

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
プロセス起動	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

3.2 提案方式の詳細

本手法では表 3.2 に示すコンポーネントを使用して各種情報を取得する．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ない事により互換性と信頼性を確保している．

そして，これらコンポーネントの組み合わせを図にしたものが図 2 である．

表 2　取得に使用するコンポーネント

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)	Windows Filtering Platform [3]

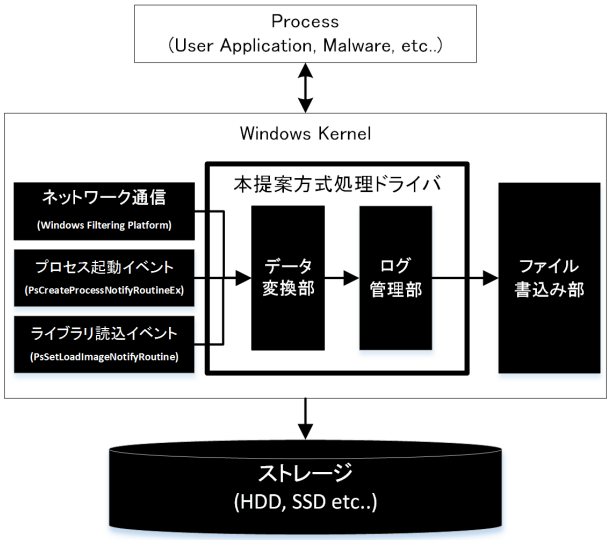

図 2　提案方式の構造

これらのうちプロセス情報の取得に関しては PsSetCreateProcessNotifyRoutineEx 及び PsSetLoadImageNotifyRoutine 関数を用い，通信に関しては Windows Filtering Platform を使用して情報の取得を行う．この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．またこれらはカーネルドライバでのみ利用が可能な API であるため，本手法ではこれらコンポーネントの呼び出しおよびログの書き出し手順を記述したカーネルドライバを対象システムに導入させることで手法を実現する．

まずプロセス情報の取得に関してであるが，これは予めドライバの内部に上記で述べた 2 関数に対応する「コールバック関数」と呼ばれる関数を用意しておき，これらに対して PsSetCreateProcessNotifyRoutineEx 及び PsSetLoadImageNotifyRoutine 関数を用いて Windows システムに登録させることで取得可能な状態にしている．これにより，プロセスが新規で起動しようとした際に事前に登録してある関数が呼び出され，ログの書き出しが行えるようになる．

続いて通信情報の取得に関してであるが，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，通信確立時を意味する FWPM_LAYER_ALE_FLOW_ESTABLISHED に判断を行うプログラムであるとしてシステムに登録し，通信が発生し呼び出された際は必要な情報を取得した後，通信の内容に関わらず別のフィルタに判断を委任させる FWP_ACTION_CONTINUE を結果として返答し，通信の通信可否の判定は他のファイアウォールに委任する構造を取る．

最後にこれらコンポーネントで収集した情報をログとしてファイルに書き出す．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

評価は次の環境で実施した．

表 3　評価に使用したソフトウェア及び機材

種別	製品名
仮想マシン	Microsoft Hyper-V
サーバ	Microsoft Windows 8.1 Pro (x86_64)
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)
開発用ソフトウェア	Microsoft Visual Studio 2013

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た．また，本提案手法を実現するカーネルモードドライバを Windows Driver Frameworks [2] 及び Microsoft Visual Studio 2013 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出来るように実装し，評価用のソフトウェアの起動する直前に開始してログを取得した．

また，評価用マルウェアは TrendMicro の表記で ”TROJ_MIRUEF.LWR”， Symantec の表記で”Trojan.Zbot” と表記されるマルウェアで，一般に ”Zeus” と呼ばれているものである．なおこのマルウェアは実験時点で C&C サーバが停止しているものを用いており，本実験による他の機器への影響は出ていない．

4.2　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．結果は下図に示す通り正常に記録できていることが確認できる．

次に下図に示す環境を用い，取得時にシステムに掛ける負荷を確認した．

まずネットワーク通信に関わる負荷として TCP,UDP そ

```
PROCESS_LAUNCH,4348,6256,¥??¥C:¥Windows¥system32¥PING.EXE,ping www.google.co.jp,,,,,,
PROCESS_MODLOAD,4348,,¥Device¥HarddiskVolume2¥Windows¥System32¥PING.EXE,,,,,,,
PROCESS_MODLOAD,4348,,¥SystemRoot¥System32¥ntdll.dll,,,,,,,
```

図 3　プロセス起動時のログ

```
PROCESS_MODLOAD,4348,,¥Windows¥System32¥rasadhlp.dll,,,,,,,
PROCESS_MODLOAD,4348,,¥Windows¥System32¥FWPUCLNT.DLL,,,,,,,
NETWORKV4,4,,,,192.168.0.39,8,173.194.126.223,0,1
NETWORKV4,4,,,,192.168.0.39,8,173.194.126.223,0,1
```

図 4　通信実行時のログ

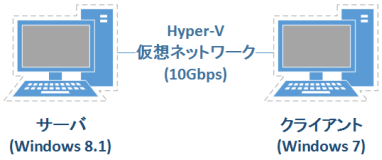

図 5　システム負荷計測時のネットワーク構成

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，10Gbps のネットワークに接続されているものとしてリンクアップする状態とした．計測は 3 回実行し数値は 3 回の平均値とした．結果を次に示す．

表 4　TCP を用いた受信方向の速度評価 (単位: KB/sec)

転送量	非適用時の速度	適用時の速度
1KB	179,131	188,062
2KB	219,033	218,233
4KB	287,228	284,590
8KB	377,493	378,167
16KB	469,127	443,881
32KB	571,569	493,265
平均値	350,597	334,366

表 5　TCP を用いた送信方向の速度評価 (単位: KB/sec)

転送量	非適用時の速度	適用時の速度
1KB	195,742	204,236
2KB	282,329	280,298
4KB	341,923	339,699
8KB	380,519	365,312
16KB	437,946	419,781
32KB	495,262	410,063
平均値	355,620	336,565

上記結果を見るに，TCP 接続において若干の速度低下が見られることが分かる．ただし，適用時の速度は受信方向で 326.53MB/sec, 送信方向で 328.68MB/sec の速度であり，一般に普及しているネットワーク回線が 1Gbps の回線 (128MB/sec) であることを鑑みると，通常利用において問題にはならないと考えられる．

表 6　UDP を用いた受信方向の速度評価 (単位: KB/sec)

転送量	非適用時の速度	適用時の速度
1KB	87,092	80,866
2KB	101,616	104,073
4KB	176,306	173,800
8KB	242,605	207,948
16KB	247,365	303,500
32KB	58,093	45,522
平均値	152,179	152,618

表 7　UDP を用いた送信方向の速度評価 (単位: KB/sec)

転送量	非適用時の速度	適用時の速度
1KB	93,028	89,432
2KB	46,833	46,392
4KB	73,324	72,374
8KB	58,990	59,515
16KB	77,095	76,429
32KB	92,510	92,363
平均値	73,630	72,751

次にシステムに掛かる負荷について Futuremark PCMark 8 による Office Benchmark を用いて評価を実施した．この評価手法は一般のオフィス環境におけるコンピュータ作業で発生するシステム負荷を想定した評価であり，本提案手法を各クライアントコンピュータに対して適用した場合の指標として判断できる．結果を次に示す．

表 8　Futuremark PCMark 8 Office Benchmark の結果

非適用時	4319
適用時	4264

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小さく，無視できる範囲であると考えられる．

最後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．この実験では表 3 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．評価中はマルウェアを起動する以外の操作を一切を行わず，10 分間放置しログを採取した．検証環境を次に示す．

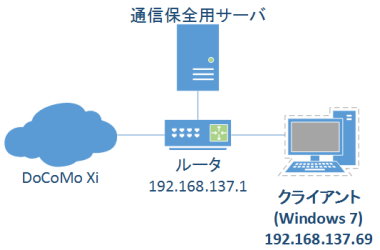

図 6　マルウェア動作取得試験時のネットワーク構成

まず，本手法にて取得したログの内容を次ページの表 9

に示す．ログは膨大な長さとなるため適宜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．

表 9　本手法によるマルウェア動作時の取得ログ

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				
2014-02:42:16:432	PROCESS_LAUNCH	2692	2716	C:¥Program Files¥Internet Explorer ¥iexplore.exe SCODEF:2716 CREDAT:267521 /prefetch:2				
2014-02:42:55:191	NETWORKV4	2692			192.168.137.69	49449	111.221.74.251	443
2014-02:42:55:284	NETWORKV4	2692			192.168.137.69	49448	111.221.74.251	443
2014-02:42:55:367	PROCESS_QUIT	2692						
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

次に，通信保全用サーバにて記録した通信内容との照合を行った．Wireshark での表示の一部と表 9 から関連する箇所のみを抽出した表を次に示す．

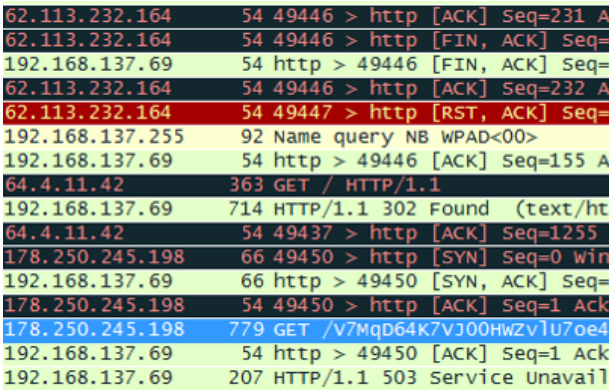

図 7　wireshark での通信表示（抜粋）

表 10　通信記録に該当するプロセス起動情報の抜粋

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 11　通信記録に該当する通信ログの抜粋

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

上記より図 7 において示されている通信のうち，送信元ポートが 49446 もしくは 49447 である通信は表 11 及び表 10 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 10 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．

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出来た．次に課題点を示す．

5.1　寄生型マルウェアの検出

本手法で取得するプロセスに関する情報は，実行ファイルにより起動したプログラムと読み込まれたモジュールの情報を記録するが，すでに起動済みのプログラムに対して悪意のあるコードを注入することで攻撃を行うマルウェアも存在する．そのようなマルウェアが実行された場合，は入本注手法でされたプログラムが不正な挙動を実施していると記録するため，注入の有無が不明な場合において誤った認識をログを通して与えてしまう可能性がある．

この問題に対しては，コードの注入を検知する処理を追加し，注入処理が実行された場合にその情報のログへの記録を可能にする必要がある．

5.2　ログの肥大化

ログは動作した分だけ記録していく為，ログが肥大化する可能性はある．確かに本手法により記録されるログの分量は，筆者が本論文執筆時に動作させた場合では平均 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 30MB ほどに肥大化することが予想される．

しかし，本手法で記録されるログは CSV 形式であるためにファイル圧縮が十分に効き，先述の実験において取得された 10.3MB のログファイルに対し zip 圧縮を実行したところ 738KB まで圧縮されることが分かっている．そのため，基本的にはログファイルによりディスクストレージが圧迫されることは，ほぼないと言える．

また長期に渡る運用を実施する場合においても，ログを 1 週間や 1 ヶ月といった期間で消去するといった運用や，ログ取得を開始するタイミングを制御することにより十分運用可能である．

5.3　SIEM 等と連携した場合における誤認識

先述の寄生型マルウェアの検出と重なる部分もあるが，本手法を SIEM に代表される統括的なログ解析ツールと連携して攻撃兆候の検知を実施した場合において，誤認識が発生し，不要なアラートや攻撃の見過ごしが発生する可能性がある．

一例として，表 9 に挙げた例の場合，マルウェアとブラウザの通信を完全に分離して判定が行えるため，特定のプロセスが特定のサーバへの接続試行を繰り返している，もしくは特定のサーバとの接続を常に張りその時間も長いといったルールにより攻撃兆候検知の一要素として使用することが出来る．

一方で寄生型マルウェアがブラウザにコードを注入し，ブラウザの通信として不正通信を実施していた場合，不正通信に関してもブラウザの通信であるということにより不正通信を見過ごし，通常の攻撃兆候検知よりも検知が遅くなる可能性が考えられる．

この問題に対しては，本手法で取得される情報を検知目

的の補助要素としてのみ用いることに加え，このような問題が発生する可能性があることを考慮に入れた検知ルールを作成すれば良い．一例として，特定のプロセスの通信であればすべて許可という判断を実施しないようにするなどが挙げられる．

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．本提案手法により取得出来る情報を活用し，攻撃の検知と情報の保全をより現場にて活用可能な形で実現する為に，現在当研究室において LIFT (Live Intelligent Network Forensics Toolkit) プロジェクト [11] [12] として筆者らにより研究が進められている．

主な活用手法の例として，通信パターンの解析により不正通信の兆候を検知した際に対象コンピュータにおいて追加情報の収集を開始するトリガーとしての活用を検討している．

また，研究室以外においても企業と連携して実製品としての活用が検討されており，現在そちらに関しても協力して研究を続行している．

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共に，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出来た．

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共に，6 章で述べた書く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

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（http://www.kaspersky.co.jp/downloads/pdf/advanced-persistent-threats-not-your-average-malware_kaspersky-endpoint-control-white-paper_jp.pdf）

[2] MSDN, "Windows Driver Frameworks" (http://msdn.microsoft.com/en-us/library/windows/hardware/ff557565(v=vs.85).aspx)

[3] MSDN, "Windows Filtering Platform" (http://msdn.microsoft.com/en-us/library/windows/desktop/aa366510(v=vs.85).aspx)

[4] MSDN, "PsSetCreateProcessNotifyRoutineEx routine" (http://msdn.microsoft.com/en-us/library/windows/hardware/ff559953(v=vs.85).aspx)

[5] MSDN, "PsSetLoadImageNotifyRoutine routine" (http://msdn.microsoft.com/en-us/library/windows/hardware/ff559957(v=vs.85).aspx)

[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,2014-CSEC-64(33), 1-8, 2014-02-27

[8] H Tirli, A Pektas, Y Falcone, N Erdogan, "Virmon: A Virtualization-Based Automated Dynamic Malware Analysis System", "INTERNATIONAL INFORMATION SECURITY & CRYPTOLOGY CONFERENCE" (http://www.iscturkey.org/iscold/ISCTURKEY2013/files/paper39.pdf)

[9] Z Liu, P Chen ,"Improved Method of packet Filtering" (http://academypublisher.com/proc/wisa09/papers/wisa09p294.pdf)

[10] Park, S., "Malware Expert: Execution Tracking",Cybercrime and Trustworthy Computing Workshop (CTC), 2012 Third (http://ieeexplore.ieee.org/xpl/abstractAuthors.jsp?arnumber=6498427)

[11] 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

[12] 比留間 裕幸, 橋本 一紀, 柿崎 淑郎, 八槇 博史, 上原 哲太郎, 佳山 こうせつ, 松本 隆, 佐々木 良一,"標的型メール攻撃に対する知的ネットワークフォレンジックのための予兆検知と対策方法の提案", DICOMO 2014 発表予定