マルウェアアナライザ Alkanet によるマルウェア解析報告 2012

大月 勇人† 若林 大晃† 瀧本 栄二† 齋藤 彰一‡ 毛利 公一†

† 立命館大学
525-8577 滋賀県草津市野路東 1-1-1
{yotuki, hwakabayasi, takimoto, mouri}@asl.cs.ritsumei.ac.jp

‡ 名古屋工業大学
466-8555 名古屋市昭和区御器所町
shoichi@nitech.ac.jp

あらまし 近年, マルウェアの脅威が問題となっている. マルウェア対策には, マルウェアの挙動を調査する必要がある. しかし, マルウェアは, 1 日に数千もの新種や亜種が出現しているため, 個々のマルウェアの調査に時間を費やすことはできない. そこで, 我々は, 仮想計算機モニタ BitVisor ベースの動的解析システム Alkanet を開発している. 今回は, Alkanet を用いて, CCCDATAset2012 に活動が記録されたマルウェアについて解析を行った. 本論文では, その調査報告について述べる.

Malware Analysis Report 2012 using Malware Analyzer Alkanet

Yuto Otsuki† Hiroaki Wakabayashi† Eiji Takimoto† Shoichi Saito‡
Koichi Mouri†

†Ritsumeikan University
1-1-1 Nojihigashi, Kusatsu, Shiga 525-8577 Japan
{yotuki, hwakabayasi, takimoto, mouri}@asl.cs.ritsumei.ac.jp

‡Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya, Aichi, 466-8555 Japan
shoichi@nitech.ac.jp

Abstract Recently, malware has become a major security threat to computers. Responding to threats from malware requires malware analysis and understanding malware behavior. However, malware analyst cannot spend the time required to analyze each instance of malware because unique variants of malware emerge by the thousands every day. We are developing Alkanet, a malware analyzer that uses a virtual machine monitor based on BitVisor. We analyzed real malware samples using Alkanet. The samples are actual instances of malware recorded in CCC DATAset 2012. In this paper, we describe the analysis reports.

1 背景

近年, マルウェアの脅威が問題となっている. マルウェア対策には, マルウェアを解析し, どのような挙動をするかを調査する必要がある. しかし, 1 日に数千もの新種や亜種が出現しているため, 1 体のマルウェアの解析に時間を費やすことができない. そこで, 比較的短時間で動作の概要を取得可能な動的解析に着目した.

しかし, 最近のマルウェアには, アンチデバッ

グと呼ばれる機能を持つもの [1, 2] や, 単一のプロセスを越えて拡散するものが多く, 解析自体が困難となっている.

上記の背景から, VMM を用いた動的解析システム Alkanet を開発している [3, 4]. 以下, 本論文では, 2 章で Alkanet の概要と構成について述べる. 3 章で実際に Alkanet 上で動作させたマルウェアの解析結果と考察について述べる. 4 章で Alkanet の今後の課題について述べる. 最後に, 5 章で本稿をまとめる.

2 Alkanet

2.1 概要

Alkanet は VMM を用いたマルウェア動的解析システムである. マルウェアのアンチデバッグを回避し動的解析を行うためには, マルウェアより高い権限で解析機構を実現する必要がある. VMM にマルウェア解析機構を実現することで, より高い権限で解析を行うことができる. これにより, 多くのアンチデバッグの手法を回避できる.

マルウェアの挙動を取得するためには, その意図の理解容易性および解析速度の観点から, 機械語レベルよりも API レベルのトレースが有効である. 特に, ユーザモードのマルウェアがシステムに影響を及ぼす動作を行うためには, システムコールを発行する必要があることから, システムコールのトレースによってマルウェアの挙動を取得する.

システムコールトレースを実現するためには, システムコールが発行される度に, その種類と引数, および戻り値を取得する必要がある. しかし, VMM は, OS の外部に存在するため, OS の API を使用できず, OS レベルの情報が取得できないという問題がある. そこで, Alkanet は, Windows が使用するメモリ領域を参照し, 独自に Windows のデータ構造の解釈を行う. VMM は, OS より高い権限で動作するため, OS のメモリ領域も含め, 仮想計算機内の全てのメモリ領域にアクセスできる.

取得したシステムコールトレースのログにより, マルウェアの挙動を分析できるが, 大規模・複雑なマルウェアであった場合, 記録されるログは人手での分析するには膨大な量となる. そこで, システムコールトレースのログをさらに分析し, マルウェアの特徴的な挙動を抽出したレポートを出力するツール群も構築している.

2.2 構成

Alkanet の全体構成を図 1 に示す. Alkanet は, VMM である BitVisor[5] の拡張機能として実装している. BitVisor は, ホスト OS を必要としないハイパーバイザ型の VMM である. Intel 製 CPU における仮想化支援機能の Intel VT(Virtualization Technology) を利用しているため, ソフトウェアのみで実現されたエミュレータや VMM に比べ, 高速に動作することが可能である. また, Windows を修正なしで実行することができる. さらに, BitVisor はハードウェアのエミュレートを行わない準パススルー型の VMM である. そのため, ゲスト OS は実マシンに搭載されているハードウェアが見られる. よって, QEMU などの特定のハードウェアをエミュレートするエミュレータや VMM に比べ, ハードウェア的特徴からマルウェアに検出されることはない.

マルウェアの実行環境であるゲスト OS には, 32bit 版 Windows XP SP3 を用いている. この環境で発行されるシステムコールをフックし, その種類や引数を取得・記録する.

Alkanet で取得したログは, ロギング用 PC から IEEE 1394 を用いて取得する. IEEE 1394 は, 接続先デバイスの物理メモリを Direct Memory Access で読み書きできる. そのため, マルウェアに検知・妨害されずにログの取得が可能である. また, ログ取得後に, ログを分析し, マルウェアの挙動を示すレポートを出力するツール群もロギング用 PC 上で動作する.

2.3 システムコールトレース

2.3.1 フック手法と対象

典型的なマルウェアの機能は, 主に以下の挙動で構成される.

- ファイルのオープン, 作成, 読み込み, 書き込み

表 1: Alkanet が監視するシステムコール

挙動	フックするシステムコールの例
ファイルの操作	NtCreateFile, NtOpenFile, NtWriteFile, NtReadFile, NtSetInformationFile, NtDeleteFile
レジストリの操作	NtCreateKey, NtOpenKey, NtQueryKey, NtSetInformationKey, NtDeleteKey, NtSetValueKey, NtQueryValueKey, NtDeleteValueKey
仮想メモリの操作	NtWriteVirtualMemory, NtReadVirtualMemory, NtAllocateVirtualMemory, NtProtectVirtualMemory
ファイルマッピングの操作	NtCreateSection, NtOpenSection, NtMapViewOfSection
ネットワークの操作	NtDeviceIoControlFile, NtReadFile, NtWriteFile
プロセスの作成/終了	NtCreateProcess, NtCreateProcessEx, NtTerminateProcess
スレッドの操作	NtCreateThread, NtSuspendThread, NtResumeThread, NtTerminateThread, NtGetContextThread, NtSetContextThread
ドライバのロード/アンロード	NtLoadDriver, NtUnloadDriver

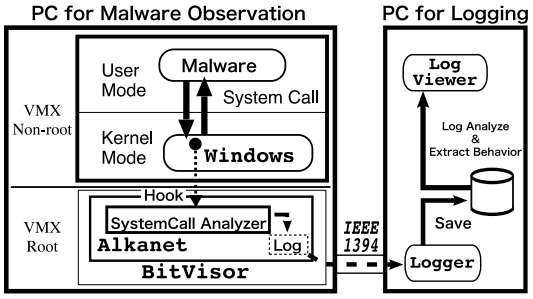

図 1: Alkanet の全体構成

- レジストリの参照, 設定
- 仮想メモリの書き込み, 読み込み, 確保, 権限の変更
- ファイルマッピングの作成, オープン, ビューの作成
- ネットワークへの送信, 受信
- プロセスの作成, 終了
- スレッドの作成, 終了, 停止, 再開, コンテキスト変更
- ドライバのロード, アンロード

具体的には, 表 1 に示すシステムコールをトレースする.

2.3.2 取得情報

Windows における実行単位は, スレッドである. また, マルウェアによるコードインジェクションによって, 通常のプロセスの中にも「悪意あるスレッド」が存在する場合がある. したがって, システムコール発行元をスレッドレベルで区別するために, プロセス ID とスレッド ID の組である Cid とイメージ名を取得する. さらに, マルウェアの挙動を調査するために, システムコールの引数と戻り値を取得する. しかし, 引数や戻り値には, ポインタや OS 固有のデータ構造が用いられることが多く, その値だけでは不十分な場合がある. したがって, これらのデータ構造を解釈し, 必要な情報を補う必要がある. 以上から, 次の情報を取得する.

- システムコール発行元の Cid とイメージ名
- システムコール番号
- システムコールの引数と戻り値
- 固有のデータ構造に対する補足情報

表 2: 今回解析した検体の内訳

検出名 (カスペルスキー)	検体数	ログエントリ数
Backdoor.Win32.EggDrop.cpc	1	73501
Backdoor.Win32.EggDrop.cpe	1	83000
Backdoor.Win32.IRCBot.jwy	2	37000
Backdoor.Win32.IRCBot.kzr	1	70000
Backdoor.Win32.IRCBot.xu	1	42000
Backdoor.Win32.Rbot.adqd	2	1435891
Backdoor.Win32.Rbot.bni	2	850000
Net-Worm.Win32.Allaple.a	5	179000
Net-Worm.Win32.Allaple.b	8	200000
Net-Worm.Win32.Allaple.e	5	194000
Net-Worm.Win32.Kido.dam.am	1	-
Net-Worm.Win32.Kido.ih	3	-
Trojan-Dropper.Win32.Injector.bslj	1	88500
Trojan.Win32.Genome.rioo	1	85000
Trojan.Win32.Genome.ujat	1	-
Trojan.Win32.Inject.achx	1	31500
Trojan.Win32.Jorik.Poebot.bt	1	-
Trojan.Win32.VBKrypt.imgt	1	80000
Virus.Win32.Virut.av	1	-
Worm.Win32.Ngrbot.jit	1	81501

3 マルウェア解析報告

Alkanet を用いて, MWS 2012 Datasets[6] の CCC DATAset 2012 内で活動が記録されている実際のマルウェア 40 体を用いて解析を行った. 文献 [7] では, 2 分程度を解析時間と設定すれば, 30 分間で動作するコードの約 90%を実行したときの解析結果が得られることが示されている. そこで, この数値を参考に Alkanet による遅延も考慮し, マルウェアを起動から 3 分間実行し, その間動作している全てのプロセスから発行されるシステムコールをトレースした. マルウェアの選別には, カスペルスキー社のアンチウィルスによる検出を基に行い, マルウェアファミリ単位で網羅するようにした. なお, ネットワークには接続していない.

3.1 データセットと各ログエントリ数

今回解析を行った 40 体のカスペルスキーにおける検出名と検体数, 記録されたログのエントリ数の表 2 に示す. “Allaple” という名称の付くものが多くなっているが, これは, そもそも CCC DATAset 2012 に記録されている検体の大半が Allaple であったためである.

今回の解析によって取得されたログサイズが最大であったのは Backdoor.Win32.IRCBot.xu (ハッシュ値 e95f7...) で 1,435,891 エントリであった. この検体では, サービスとして起動した自身のプロセス内に大量のスレッドを作成する他, ipconfig.exe を何度も起動しては終了するという挙動を行う. スレッドやプロセスを起動する場合, 実行するファイルのメモリへマップ, 初期スレッド作成など, 起動のための準備を行うため, 短時間で大量のシステムコールが発行される.

表 3: 挙動別の検体数

挙動	検体数 (割合)	種類数 (割合)
動作した検体	33 (82.5%)	17 (85.0%)
ファイルを作成	24 (60.0%)	10 (50.0%)
ドライバを作成	2 (5.0%)	2 (10.0%)
新たにプロセスを起動	22 (55.0%)	11 (55.0%)
コードインジェクション	10 (25.0%)	8 (40.0%)
自分自身のファイルの削除	15 (37.5%)	5 (25.0%)
サービスとして起動	15 (37.5%)	4 (20.0%)
Run キーの設定	5 (12.5%)	5 (25.0%)
LocalServer32 キーの設定	21 (52.5%)	5 (25.0%)
名前付きパイプによるプロセス間通信	21 (52.5%)	8 (40.0%)
/etc/host を変更	2 (5.0%)	2 (10.0%)
ファイアウォールの設定の変更	5 (12.5%)	5 (25.0%)
ネットワークへの接続	4 (10.0%)	3 (15.0%)

Net-Worm.Win32.Allaple.a, Net-Worm.Win32.Allaple.b など, “Allaple” という名前で検出されているマルウェアでも, 何度もスレッドの作成と終了が繰り返されていることが観測された. このため, これらのマルウェアの動作ログのエントリ数も比較的大きいものとなっている.

また, ログエントリ数が “-” となっている検体では, マルウェアが実行できない, あるいは実行開始直後に停止した場合などにより, マルウェア自身の挙動が全く観測されたかったものである.

3.2 挙動別の内訳

表 3 に今回の解析でマルウェアによく見られた挙動の一覧を示す. 種類数は, カスペルスキーの検出名を基に算出した.

3.2.1 ファイルをドロップする検体

ファイルを新たに作成するマルウェアが多く見られたが, その多くが以下の場所にファイルを作成する.

- `\WINDOWS\system`
- `\WINDOWS\system32`
- `\WINDOWS\system32\drivers`
- `\Documents and Settings\...\Application Data`

上記に作成されたファイルの多くは, Run キーや LocalServer32 キー, サービスを登録する時にパスが指定されていた. したがって, これらに置かれるファイルは, マルウェア自身のコピーであると考えられる. Run キーに設定し, システム起動時に自動的に起動するよう設定する検体は 5 体存在した. また, “Allaple” の名称で検出される検体を中心に LocalServer32 キーに登録するものも 21 体存在する. サービスを利用する検体 12 体はおり, このうち, Net-Worm.Win32.Allaple.b が 5 体, Net-Worm.Win32.Allaple.e が 4 体である. 15 体の自分自身の実行ファイルを削除する検体は, 上記のディレクトリにファイルを作成する検体と共通する.

作成されるファイルは, csrsc.exe や svhost.exe, nservice.exe, sysdrv32.sys など, 正規の Windows プロセスと似ているものや存在しても違和感のないものが多い. また, 登録されたサービスも “Network Windows Service”, “Windows Spool Services” など, 正規のサービスとして存在しても違和感のない名前になっている.

また, “Allaple” という名称を含むマルウェアは, `\WINDOWS\system32` にファイルを作成するだけでなく, `\Program Files\Common Files`

や, \WINDOWS\system32\oobe\ 以下など, 様々の位置な位置に .htm ファイルや .exe ファイルを作成していた. ここで作成されたファイルは, rsewzjqn.exe や njnrhctz.exe など, ランダムに生成されたようなファイル名となっている.

ファイルを作成した検体のなかでも, Backdoor.Win32.IRCBot.kzr (09851...), Trojan.Win32.Inject.achx (6acb0...) という二種のマルウェアは, sysdrv32.sys というドライバを \WINDOWS\system32\drivers にドロップする. これらのマルウェアは, その後, レジストリにこのドライバを登録するための項目を作成し, Play Port I/O Driver という名称で, カーネルモードドライバとして設定する. しかし, スタートアップの種類が手動に設定されており, 肝心の NtLoadDriver も発行されなかった. したがって, このドライバはなんらかの条件下でロードされるものであると考えられる.

3.2.2 プロセスを作成する検体

22 体の検体が, 最初に起動したマルウェアのプロセスの他に新しくプロセスを起動する. 例えば, Backdoor.Win32.EggDrop.cpc (54932...) などのいつかの検体は, 自身を再起動している. これは, 一般的なデバッガのアタッチできるプロセスが一つのみであるため, 自身を再起動することで追跡を回避しようという試みである. また, 多くの検体は, まず前述のようにシステムディレクトリに正規プロセスのような名前を持つファイルを作成する. その後, そのファイルをプロセスあるいはサービスとして起動し, 最初のプロセスは終了する. これにより, 正規プロセスのような名前を持つ悪意あるプロセスが動作するため, ユーザの目を誤魔化しやすくなると考えられる.

Backdoor.Win32.IRCBot.xu (e95f7...) に起動される ipconfig.exe は, このプログラムは本来の用途から, これはネットワークの状況を確認する挙動である. 一般的な API を使用しないことで, 解析者によるフックを逃れ, 解析を困難にする目的があると考えられる.

3.2.3 コードインジェクションを行う検体

10 体のマルウェアが別の正規のプロセスに対してコードインジェクションを行っていた. インジェクションされるプロセスに多いものを以下に示す.

- explorer.exe
- rundll32.exe
- winlogon.exe

これらのプロセスに対し, NtWriteVirtualMemory によるメモリ空間への書き込みと, NtCreateThread によるスレッドの作成が行われていた. これは, WriteProcessMemory API と CreateRemoteThread API の組み合わせたコードインジェクションの挙動である.

この他, Virus.Win32.Virut.av (78ab0...) では, 上記の他に services.exe, lsass.exe, nvsvc32.exe, svchost.exe, spoolsv.exe に対してコードインジェクションを行っていた. こちらの場合, メモリ書き込みだけであった.

3.2.4 名前付きパイプを用いてプロセス間通信する検体

21 体のマルウェアが名前付きパイプを用いて, 他のプロセスと通信している. 特に \lsaprc を用いて lsass.exe と通信するものや, \net\NtControlPipe10 を用いて services.exe と通信するもの, \wkssvc を用いて svchost.exe と通信するものなどがある.

3.2.5 ネットワークに接続する検体

今回の解析ではネットワークに接続していないが, 4 体のマルウェアがネットワークに接続する挙動を行った. これらは, 外部からの通信を待ち受けると同時に, 同一ネットワークに存在し得る IP アドレスに対して, パケットを送信し応答があるか調べていた. Windows では, ネットワークに接続していなくても, Automatic Private IP Addressing の機能によって, 169.254.1.0 などのリンクローカルアドレスが割り当てられる. そのため, 例えば, Trojan.Win32.Inject.achx として検出された検体 (6acb0...) では, 169.0.0.1, 169.0.0.52, 169.0.0.101,

... などへパケットを送信している．このパケット送信の挙動は, 1700 回ほど観測された．なお, この時の接続先ポート番号は 48385 であった．このポート番号や IP アドレスへのパケット送信の仕方には他の検体にも共通するため, 共通するモジュールが使用されているのではないかと考えられる．

また, `\WINDOWS\system32\drivers\etc\hosts` を変更する検体も 2 体存在する．これには, Windows Update やアンチウイルスソフトの更新を妨害する目的があることが考えられる．さらに, Firewall の設定を変更し, 信用できるアプリケーションに, いくつかの登録が追加する検体が 5 体存在した．これらのマルウェアは, 自分自身の実行ファイルではなく, コードインジェクションする正規プロセスを追加する．これは, マルウェア自身ではなく正規のプロセスを登録しておくことで, ユーザが後でリストを見た際に問題がないように見せること目的であると考えられる．

3.3 実行できなかった検体・実行を停止させられた検体

Alkanet 上で実行できなかった検体がいくつかある．例えば, Net-Worm.Win32.Kido.ih として検出されていたものは, すべて DLL ファイルであり, 単体では実行不可能であった．

マルウェアの起動開始直後に強制終了されてしまう検体も存在した．Trojan.Win32.Jorik.Poebot.bt (84114...) では, NtCreateProcessEx によってマルウェアのプロセスが誕生した直後に, 親プロセスである explorer.exe に NtTerminateProcess を発行され, なんの活動も観測されずに終了している． また, Net-Worm.Win32.Allaple.e として検出される検体の一つ (a4b58...) では, drwtsn32.exe が起動し, マルウェアのプロセスを終了させていた．

Backdoor.Win32.IRCBot.jwy として検出される二体 (4ead4..., 9e366...) では, Windows のデータ実行防止機能が起動し, 動作を停止させられていた．しかし, 停止させられたのは, マルウェアにコードインジェクションされた explorer.exe だけであり, サービスとして起動されたマルウェアの方は, 停止されておらず動作を継続していた．

実行後, Backdoor.Win32.Rbot.adqd (66e99..., 3690e...), Virus.Win32.Virut.av (78ab0...) という検体はブルースクリーンとなり, Windows が強制終了してしまった．ブルースクリーンになるということは, カーネルあるいはドライバで問題が発生したということである．Windows が強制したため, ブルースクリーンの直前の挙動のログを正しく記録できなかった． 観測できた範囲でのこれらの共通点は, winlogon.exe, services.exe, lsass.exe などにコードインジェクションを行っていたことまでである．

4 今後の課題

今回の解析で, 数多くの検体を短時間で解析するには, 手動の作業が多く, 自動化が不十分であると実感した．現在の Alkanet では, マルウェアを実行すると実際に Windows 環境を書換えれてしまう．また, Alkanet にはスナップショット機能などは実装しておらず, 自力で環境を復元することができない．これまでは, 別途起動前のイメージを予め取っておき, 丸ごと上書きすることで対処してきた．そこで, 表らの研究 [8] のように, VMM 側で, ゲスト OS 内で発生するディスクに対する IO をフックすることを考えている．ディスクへの IO をフックすることで, 実際のディスクを変更させないようにし, 復元の手間を減らすことを目指す．

また, 現在の Alkanet は, ネットワークを用いた挙動を十分に解析できない．ネットワークを用いた通信を解析するためには, 実際に別のサーバやコンピュータに攻撃させないように配慮する必要があるためである．現在の Alkanet にはパケットのフィルタリングや制御を行う機能は実装していない．対応として, ハニーポットなどと連携し, 実際に通信を行わせないようにする必要があると考えている．

さらに, システムコールのログからマルウェアの挙動の抽出にも時間がかかる．ある程度の自動化を行ってはいるが, 現在そのアルゴリズムは独自の経験則を基にしたものであり, 効率がよくない．そこで, 既存のシステムコールの

ログを用いた異常検知やマルウェアのクラスタリングを行う手法などを用いることを考えている. 例えば, Anubis の結果を用いてマルウェアのクラスタリングを行う研究 [9] が行われている. 今後, このような既存の手法に Alkanet のログが利用できるか評価を行う.

5 まとめ

本論文では, 仮想計算機モニタを用いて, マルウェアの動的解析を実現する “Alkanet” について述べた. さらに, Alkanet を用いて, CCC DATAset 2012 に記録されたマルウェアについて解析を行い, その結果について述べた.

プロセスやスレッドを数多く作成するマルウェアは, ログのサイズが大きくなる. これらの挙動では, 短時間で大量のシステムコールが発行されるためである. 解析した中で 60%のマルウェアが, ファイルを新たに作成する. また, 作成されたファイルの多くは, 自動起動のためのキーやサービスに登録されるため, マルウェア自身のコピーであると考えられる. また, ドライバをドロップするものもいたが, NtLoadDriver は発行されなかった.

25%のマルウェアは, 正規プロセスにメモリ空間への書き込みとスレッドの作成を行う. 今回の解析ではネットワークに接続していなかったが, 10%のマルウェアがネットワークに接続し, 外部からの通信を待ち受けると同時に, ネットワーク内の IP アドレスに対して応答があるか調べるような挙動が見られた. さらに, 実行が停止させられてしまったマルウェアの中には, データ実行防止によるものや, ブルースクリーンになり, Windows ごとが強制終了してしまうものも存在した.

参考文献

[1] N. Falliere: “Windows Anti-Debug Reference,” http://www.symantec.com/connect/articles/windows-anti-debug-reference (2007, Last accessed, July 2012).

[2] M. V. Yason: “The Art of Unpacking,” In Black Hat USA 2007 (2007).

[3] 大月勇人, 毛利公一: “仮想計算機モニタによるマルウェアの監視,” コンピュータセキュリティシンポジウム 2010(CSS2010) 論文集, pp. 225–230, 情報処理学会 (2010).

[4] 大月勇人, 瀧本栄二, 樫山武浩, 毛利公一: “マルウェア挙動解析のためのシステムコール実行結果取得法,” コンピュータセキュリティシンポジウム 2011 論文集, 第 2011 巻, pp. 95–100 (2011).

[5] T. Shinagawa, H. Eiraku, K. Tanimoto, K. Omote, S. Hasegawa, T. Horie, M. Hirano, K. Kourai, Y. Oyama, E. Kawai, K. Kono, S. Chiba, Y. Shinjo and K. Kato: “BitVisor: a thin hypervisor for enforcing i/o device security,” In Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, pp. 121–130, Washington, DC, USA (2009), ACM.

[6] MWS2012 実行委員会: “研究用データセット MWS 2012 Datasets について,” http://www.iwsec.org/mws/2012/about.html#datasets.

[7] 青木一史, 川古谷裕平, 岩村誠, 伊藤光恭: “動的解析における検体動作時間に関する検討,” コンピュータセキュリティシンポジウム 2010(CSS2010) 論文集, pp. 543–548 (2010).

[8] 表祐志, 品川高廣, 加藤和彦: “仮想マシンモニタによる透過的ネットワークブート方式,” 情報処理学会論文誌コンピューティングシステム（ACS）, Vol. 4, No. 4, pp. 228–245 (2011).

[9] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Krügel and E. Kirda: “Scalable, Behavior-Based Malware Clustering,” In Network and Distributed System Security Symposium (2009).