トラヒック解析と Virtual Machine Introspection の連携によるボット検知方式

川口 信隆　　大河内 一弥　　仲小路 博史
鬼頭 哲郎　　重本 倫宏　　寺田 真敏

(株)日立製作所　システム開発研究所
〒212-8567 神奈川県川崎市幸区鹿島田 890

概要：本論文では，ネットワーク型 IDS(NIDS)によるトラヒック解析と，Virtual Machine Introspection により端末から取得したプロセスのコンテキスト情報を連携させたボット検知方式を提案する．従来の NIDS は Command and Control 通信(C&C 通信)と感染活動の両方を行っている疑惑がある端末をボット感染端末と判定する．しかしこの手法には，場合によって誤り検知や検知見逃しが発生する可能性がある．例えば，ボットとは無関係な別々のプロセスが C&C 通信や感染活動に類似した振る舞いを行う端末は誤検知されやすい．この問題を解決するために，提案方式はプロセスのコンテキスト情報を NIDS に与えることで，プロセス粒度での検知を実現する．例えば C&C 通信と感染活動を同一プロセスが行う場合，端末はボットに感染していると判断する．Xen 上でのプロトタイプ実装と CCC DATAset 2009 で指定された検体を用いた評価実験により，提案手法がボットを検知することを確認した．

キーワード：ボット，検知，Virtual Machine Introspection

Enhancing Network-based Bot Detection using Virtual Machine Introspection

Nobutaka Kawaguchi　Kazuya Okochi　Hirofumi Nakakoji
Tetsuro Kito　Tomohiro Shigemoto　Masato Terada

Systems Development Laboratory, Hitachi Ltd.
890 Kashimada, Saiwai-ku, Kawasaki, Kanagawa, 212-8567 Japan

Abstract: In this paper, we propose a bot detection method that combines traffic analysis by Network based IDS (NIDS) and process context information obtained from monitored machines by using Virtual Machine Introspection. Some existing NIDS determine a machine is infected if it is suspected of performing both of the Command and Control(C&C) communication and infection activities. This approach, however, has possibilities to generate false positives and negatives. For example, a machine that simultaneously runs two benign processes which perform C&C communication-like and infection-like activities respectively could be falsely detected. To address this problem, the proposed method enables NIDS to achieve process-grained detection by adding the context information of the processes that perform network activities. For example, this method declares a machine is infected when the machine has a process that appears to perform both of C&C communication and infection activities. Through experiments using a prototype implementation on Xen and a bot sample specified by CCC DATAset 2009, we confirm that the proposed method has a capability to detect bots appropriately.

Keywords: Bot, Detection, Virtual Machine Introspection

1　はじめに

近年のボットの大量発生に伴い，高性能なボット検知技術の確立が急務となってきている．従来技術であるネットワーク型 IDS（NIDS）は，トラヒック解析によりボットのネットワーク活動を検知する．NIDS はネットワーク内の複数端末を同時に長期間監視できるため，複雑な解析や統計的手法を適用できるという利点がある[1]．しかしその一方で，端末より細かい粒

度での解析ができないため，通常通信に類似した様々なネットワーク活動を行うボットに対して，誤検知や検知見逃しが発生しやすい．例えば BotHunter[2]は IRC 通信と不審なメール送信を同時に行う端末をボットと見なすが，ユーザが IRC チャットを行いながら大量のメール送信をする場合に誤検知が発生する恐れがある．

我々は，ネットワーク活動を行っているプロセスに関連する情報を NIDS に与え, NIDS によるプロセス粒度での検知を可能とすることで，この問題を解決することを考える. 例えば, IRC 通信とメール送信を同一のプロセスが行っていることが判れば，端末がボットに感染している可能性は高いと判断できる．反対に，全く関係性の無い別々のプロセスが IRC 通信とメール送信を行っている場合は，ボットに感染している可能性は低いと言える．

本論文では，監視対象端末上でネットワーク活動を行っているプロセスのコンテキスト情報（プロセスを端末内で一意に識別するプロセス ID, 端末内での各プロセスの関係性を示すプロセスツリーなど）と NIDS のトラヒック解析結果を連携させて検知を行う方式を提案する．提案方式は，特定のネットワーク活動を行っているプロセスのコンテキスト情報が，予め定義したルールに適合する場合に，監視対象端末はボットに感染していると判定する．

本方式では，ボットに感染している端末からコンテキスト情報を安全に取得するために，仮想環境上で管理ドメインからゲストドメインのメモリの状態を監視する技術である Virtual Machine Introspection(VMI)[3]を用いる．このため，監視対象端末はゲストドメインとして動作する．管理ドメインは，ゲストドメインから外部へ送信されるパケットのヘッダにパケット発信元のプロセス ID を埋め込む．また NIDS に対してプロセスのコンテキスト情報を提供する．NIDS は，監視対象端末上で特定のネットワーク活動を行っているプロセスのコンテキスト情報を管理ドメインより取得し，これを元にボット判定を行う．

サイバークリーンセンター が提供する CCC DATAset 2009[4]で指定された検体を用いた評価実験を通じて，提案方式がボットを検知できることを実証した．

2 関連研究

2.1 ボット検知技術

ボットネットは，攻撃者に制御を奪われた端末（ボット）により構成されるネットワークである[5]. 端末はマルウェアの実行や脆弱性攻撃を受けることでボットに感染する．ボット感染端末は IRC サーバなどを介してボット管理者（攻撃者）から命令を定期的に受け取る Command and Control 通信（C&C 通信）を行い，命令に基づいてスパムメールの送信や他端末への攻撃といった感染活動を行う．

ボット感染端末の検知技術の 1 つとして，NIDS によるトラヒック解析がある．ホスト型 IDS と違い NIDS は専用のネットワーク装置上で動作し，ネットワーク内の複数端末を同時に長期間監視できるので，複雑な解析や統計的な手法を適用しやすい．例えば BotSniffer など[1][6][7]は，同一のボットネットに属するボットは C&C 通信や感染活動を同期して行なう場合が多いという点に着目する．そして，同一のネットワーク活動を同期して行なう端末群をボットと判断する．但し，定期的にサーバにアクセスするアプリケーションが複数の端末にインストールされている場合などに，誤り検知が発生する可能性がある．

同じく NIDS である BotHunter[1]は，C&C 通信を行った後に感染活動を行うというボットの活動シーケンスを検知に利用する. C&C 通信と疑わしい活動を行ってから一定時間内に感染活動と疑わしい活動を行なう端末はボットに感染していると判断する．

しかし BotHunter には以下の 3 つの問題点がある. 第一に, ユーザが IRC チャットなど C&C 通信に分類されやすい通信をしながら，メール送信など感染活動と疑われる可能性があるネットワーク活動を行なう場合，ボット感染端末として誤検知される可能性がある．第二に，ボットが C&C 通信を終了してから一定以上の時間が経って感染活動を開始した場合，検知することができない．第三に，感染活動に先立って端末内で行われる準備活動を検知できない．例えば，ボットの中にはメール送信活動に先立って端末内のファイルをスキャンしてメールアドレスを取得するものがあるが，トラヒック解析ではこの活動を発見できない．提案方式は，トラヒック解析と VMI を併用することでこれらの問題の解決を図る．

2.2 Virtual Machine Introspection

VMI は Xen[13]などの仮想環境上で管理ドメインからゲストドメインが使用しているリソース（メモリなど）にアクセスして，ゲストド

メイン上で行なわれている活動の監視や制御を行なう技術である．管理ドメインはゲストドメインから Hypervisor により隔離されるため，ゲストドメイン上からの VMI の妨害は不可能である．このため，セキュリティソフト（AV，ホスト型 IDS，端末内ファイヤーウォールなど）を停止させるマルウェアに対抗する技術として注目されている[8]．VMI により直接取得できる情報はメモリダンプなど低レベルなものであるため，そのままでは扱いが難しい．このため，低レベル情報を解析して高レベル情報（プロセス構造体など）を抽出するライブラリが開発されている[9]．

3　提案方式

3.1　概要

提案方式は，トラヒック解析に加えてプロセスのコンテキスト情報を検知に用いることで，既存研究で発生する誤り検知や検知見逃しの問題を解決する．

提案方式は，監視対象端末上に C&C 通信や感染活動と疑われるネットワーク活動を行っているプロセスが存在し，それらのプロセスのコンテキスト情報が予め定義されたルールに適合する場合に，端末はボットに感染していると判定する．コンテキスト情報をボット感染端末から安全に取得するために VMI を用いる．このため，監視対象端末は仮想環境上でゲストドメインとして動作することを前提する．

図 1 に本方式の概要を示す．VMI を行う管理ドメイン上には，メモリ解析モジュール，プロセス ID（PID）挿入モジュール，コンテキスト情報提供モジュールが存在する．ネットワーク活動を検知する NIDS 上には，C&C 通信検知モジュール，感染活動検知モジュール，総合判定モジュールが存在する．

以下に検知の流れを述べる．ゲストドメイン内のプロセスが IP パケットを外部に送信するとき，パケットは Hypervisor を経由して管理ドメインに到達する．このとき，管理ドメイン上の PID 挿入モジュールは，メモリ解析モジュールをからパケット送信プロセスの PID を取得して IP パケットヘッダに埋め込む．その後パケットは外部へ送出される．次に NIDS 上で動作する C&C 通信検知モジュールと感染活動検知モジュールは，ゲストドメインが送信したパケットをキャプチャして，解析を行う．そして，C&C 通信または感染活動と疑わしいネットワーク活動が検知された場合，パケット送信元プロセスの PID を総合判定モジュールに通知する．総合判定モジュールは，通知されたプロセスのコンテキスト情報をコンテキスト情報提供モジュールに問い合わせる．コンテキスト情報提供モジュールはメモリ解析モジュールからコンテキスト情報を取得し，総合判定モジュールに返答する．総合判定モジュールは，ネットワーク活動を行っているプロセスのコンテキスト情報がルールと適合した場合にゲストドメインはボットに感染していると判定する．

以下，各モジュールの詳細を説明する．

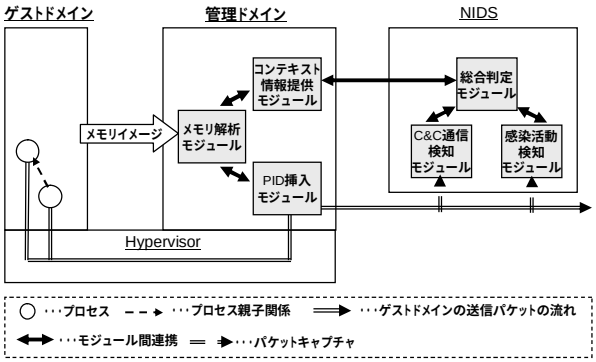

図 1 提案方式の概要

3.2　メモリ解析モジュール

メモリ解析モジュールはゲストドメインのメモリイメージを取得し，プロセスごとに以下のコンテキスト情報を抽出する．

- プロセスの PID
- プロセスが開くネットワークソケット
- プロセスの親プロセスの PID
- プロセスがアクセスするファイル

各情報の利用用途，実装方法については後述する．情報取得は他モジュールから要求を受けた際に行われる．

3.3　PID 挿入モジュール

PID 挿入モジュールは，ゲストドメインが送信した IP パケットをキューイングする．次にメモリ解析モジュールから各プロセスが開いているネットワークソケットのリストを取得する．そしてパケット送信元プロセスの PID を IP ヘッダに PID を書き込み，外部に送出する．

IP ヘッダ内に PID を書き込む方法はいくつか考えられるが，提案方式では 16bit 長の ID フィールドに PID の下位 16 bit を書き込む．ID フィールドはパケットのフラグメントが起きた場合にパケット受信元で再構築する際に用いられる．現状のインターネット上ではフラグメン

トが起きる確率は 1%以下であり，このフィールドにPIDを上書きした場合でも通信に対する影響は無い[10].

3.4 C&C 通信検知モジュール

C&C 通信検知モジュールは，C&C 通信と疑わしい通信を検知する. C&C 通信に用いられる通信方式やその検知方法は様々であるが，本方式で想定する具体例を表 1 に示す.

表 1 C&C 通信と検知方法の例

方式	説明	検知方法
IRC	攻撃者がログインしている IRC サーバにアクセスして，命令を受け取る.	検知対象端末が外部端末に接続し，特有の文字列（PONG, JOIN，NICK など）が含まれるパケットを送信することを検知する.
P2P	P2P ネットワークに参加して，攻撃者から命令を受け取る[11][12].	検知対象端末が多数の外部端末に接続すると共に，多数の接続を受けることを検知する.

C&C 通信を検知した場合，パケットの ID フィールドから活動元 PID を特定して，総合判定モジュールに通知する.

3.5 感染活動検知モジュール

感染活動検知モジュールは，感染活動と疑わしい通信を検知する．ボットが行う感染活動やその検知方法は様々であるが，本方式で想定する具体例を表 2 に示す.

表 2 感染活動と検知方法の例

感染活動	説明	検知方法
メール送信	自ネットワーク外にあるメールサーバに接続し，SPAM メールやフィッシングメール，マルウェア添付メールを送信する.	検知対象端末が，自ネットワーク外の端末の宛先ポート 25 番へ TCP 接続することを検知する.
アドレススキャン	ランダムに選択された IP アドレスや一定範囲内にある IP アドレスに対して，連続かつ高速に接続試行を行う.	検知対象端末の外部端末への接続試行が，高確率で失敗することを検知する.

感染活動を検知した場合，パケットの ID フィールドから活動元 PID を特定して，総合判定モジュールに通知する.

3.6 総合判定モジュール

総合判定モジュールは，C&C 通信検知モジュールと感染活動検知モジュールから通知された PID のコンテキスト情報がルールに適合するとき，ボット感染を検知する．提案方式が用いるルール内容を表 3 に示す．尚，C&C 通信検知モジュールと感染活動検知モジュールが通知したプロセスをそれぞれ Pc，Pi とする.

表 3 判定ルール

ルール名	内容
ルール 1	Pc と Pi は同一プロセスである.
ルール 2	Pc が Pi の先祖プロセスである.
ルール 3	Pc が，端末内で感染活動の準備活動を行う
ルール 4	Pc の子孫プロセスが，端末内で感染活動の準備活動を行う.

ルール 1 とルール 2 は，C&C 通信を行うプロセスと感染活動を行うプロセスとの関係性を元にボット判定を行う．ルール 1 は，同一プロセスが C&C 通信と感染活動を行っていることを意味する．ルール 2 は，C&C 通信を行ったプロセスが生成した子プロセスや孫プロセスが感染活動を行うことを意味する. C&C 通信と感染活動が全く関連性の無いプロセスにより行われていた場合，ボットである可能性は低いと考えられる．但し，ボットが検知を逃れるために，意図的に C&C 通信と感染活動を全く関係性の無いプロセスで実行する可能性がある．この問題については 5 章で議論する.

一方，ルール 3 とルール 4 は，感染活動に先立ってボットが端末内で行う準備活動を検知する. C&C 通信を行うプロセスもしくはその子孫プロセスが感染活動に結びつく可能性がある準備活動を行っている場合，プロセスはボットであると判断する．このため，実際に感染活動を行う前の検知が可能である.

準備活動の具体例の１つとして「メール送信先アドレス収集のためのファイル検索」がある. メール送信を行うボットの中には，ブラウザキャッシュ，メールフォルダ及びその他のファイルにアクセスしてメールアドレスを収集するものがある．このため，これらのファイルに多数アクセスしているプロセスは準備活動を行っていると判断できる.

ルール 1 以外のルールを用いた判定を行うには，コンテキスト情報提供モジュールからコンテキスト情報を取得する必要がある．表 4 に，総合判定モジュールがコンテキスト情報提供モジュールから取得する情報と，情報を取得するタイミングを示す.

表 4 コンテキスト情報の取得

ルール名	取得する情報	タイミング
ルール 2	Pi の先祖プロセスのリスト	Pi が通知されたとき.
ルール 3 ルール 4	Pc とその子孫プロセスのアクセスするファイルのリスト	Pc が通知されてから，定期的に.

3.7 コンテキスト情報提供モジュール

コンテキスト情報提供モジュールは総合判定モジュールからの要求を受けると，メモリ解析モジュールからコンテキスト情報を取得する．そして，取得した情報を要求に合うように加工して返答する．

4 評価実験

提案方式を実装し，CCC DATAset 2009 の検体を対象に評価実験を行った．以下，実装，検体，ボット実行環境，実験結果について記す．

4.1 実装

表 5 に提案方式の実装環境を示す．仮想環境として Xen を用いた。ゲストドメイン上では Windows XP SP2 が動作する．

プロセスのコンテキスト情報取得は以下のように行う．メモリ解析モジュールは，先ず XenAccess[9]を用いてゲストドメインのメモリを管理ドメイン上にマウントする．次に，マウントされたメモリをメモリフォレンジックツールである Volatility Framework[14]を用いて解析してコンテキスト情報を取得する．

各モジュールの大部分は Java で記述した．IP ヘッダへの PID 書き込みには libipq を用いた．

表 5 実装

OS	CentOS5.2
仮想環境	Xen3.2
VMI	XenAccess0.5, Volatility Framework1.3
開発言語	JDK1.6
PID 書き込み	Libipq

4.2 検体

実験に用いる検体は，Virus.Win32.Virut.n[16]である．検体は起動後 IRC サーバにアクセスしてマルウェアのダウンロード命令を受ける．次に, 検体は指示された Web サーバからマルウェアをダウンロードして実行する．実行されたマルウェアは外部メールサーバにアクセスし，メール送信を行う．

4.3 ボット実行環境

検体のネットワーク活動を再現するために，仮想環境上に IRC サーバ，Web サーバ，DNS サーバが動作するドメイン（サーバドメイン）を構築した．ゲストドメインのデフォルトゲートウェイはサーバドメインに設定されている．ゲストドメイン，サーバドメイン，管理ドメインは仮想ブリッジを介して接続されている．

DNS サーバは全ての名前解決要求に対してサーバドメインの IP アドレスを返す．IRC サーバはボットからアクセスを受けるとマルウェアのダウンロード命令を送る．Web サーバはボットにマルウェアを配布する．パケットへの PID 書き込みは，仮想ブリッジにパケットが入力される時に行われる．IDS はサーバドメイン上で動作して，ゲストドメインからサーバ宛のパケットをキャプチャして解析する．

4.4 実験結果

図 2 にボット起動により生じるプロセスツリーの一部を示す．ボットは起動直後に PID=472 の WINLOGON プロセスに感染する．PID=472 は IRC サーバにアクセスし， C&C 通信検知モジュールに検知される．以後，総合判定モジュールはPID=472 とその子孫プロセスが開いているファイルリストを数秒おきに管理ドメインから取得するが，感染活動の準備活動（メールアドレスの収集）は検知されなかった．

次に，ボットはマルウェアを WEB サーバからダウンロードして実行する．そして IRC サーバへのアクセスから 5 分後に，PID=180 の services プロセスが gmail.com など外部メールサーバへの SMTP 接続試行を開始し，感染活動検知モジュールに検知される．このとき総合判定モジュールは services プロセスの祖先プロセスリストを取得し，PID=180 は PID=472 の子孫プロセスであることが明らかになる．よって表 3 のルール 2 に基づき，ゲストドメインはボットに感染していると判定される．尚，PID=180 以外にも複数のプロセスが SMTP 接続試行を行い，検知される．

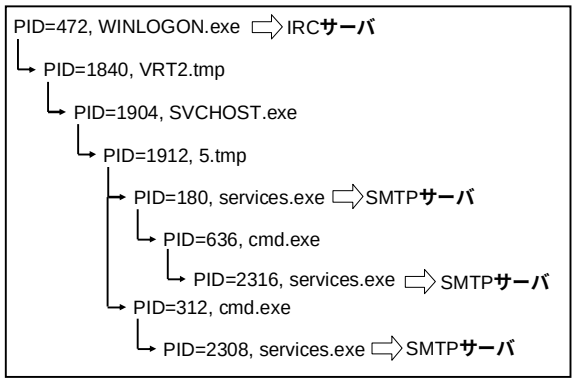

図 2 検体のプロセスツリー

5 考察と今後の課題

BotHunter と比較して，提案方式は以下の 3 つの利点がある．第一に，異なるアプリケーションが C&C 通信と感染活動に類似した活動を別々に行っている場合，提案方式では誤り検知

が発生しない．第二に，提案方式では C&C 通信と感染活動が発生する時間が離れている場合に検知見逃しが発生するが，提案方式では両活動を行うプロセス間に関連性がある限り，検知可能である．第三に，BotHunter では検知できない準備活動を，提案方式では検知できる

その一方で，提案方式ではボットが C&C 通信活動と感染活動を別々の子孫プロセスで行った場合や CreateRemoteThread API を呼び出して別プロセスに行わせた場合に，検知見逃しが発生するという問題がある．このためプロセス間の関連性については，より詳細な解析が必要である．また本実装では Volatility Framework を用いて，要求がある度にメモリ解析を行ったが，この方法ではメモリ解析を行う間に生成・実行・終了したプロセスのコンテキスト情報を確実に取得できない可能性がある．この問題を解決するには，ゲストドメインに API フックを仕掛け，特定の API 呼び出しがあるたびに管理ドメインに通知が行く仕組みが必要である[15].

また，IRC チャットやメール送信，アドレススキャンなど複数のネットワーク機能を備えた統合型アプリケーションを，提案方式は BotHunter と同様に誤り検知する可能性がある．

6 おわりに

ボットネットの蔓延に伴い，高性能なボット検知技術の確立が急務となっている．ネットワーク活動に着目したボット検知方式は有効なアプローチの１つであるが，誤り検知や検知見逃しが発生する場合がある．

本論文では，この問題を解決するために，NIDS のネットワーク活動解析結果と VMI により取得したプロセスのコンテキスト情報を連携させることでプロセス粒度でのボット検知を実現する方式を提案した．そして，Xen 上に提案方式を実装して，CCC DATAset 2009 で指定された検体を検知できることを示した．

今後はより多くの検体と正規アプリケーションを用いて，検知性能を測定する．また，コンテキスト情報を取得する仕組みを改良して，複雑なプロセス間連携を行うボットを検知できるようにする予定である．

謝辞

本研究は独立行政法人情報通信研究機構から委託を受けた「マルウェア対策ユーザサポートシステムの研究開発」の成果の一部を含みます．本研究を進めるにあたって有益な助言と協力を頂いた関係者各位に深く感謝いたします．

参考文献

[1] G.Gu, et al., “BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic”, In Proc. of NDSS’08, 2008

[2] G.Gu, et al, “BotHunter: Detecting malware infection through ids-driven dialog correlation”, In Proc. of the 16th USENIX Security Symposium, 2007.

[3] K.Nance, et al, “Virtual Machine Introspection: Observation or Interference?”, IEEE Security and Privacy Magazine, Vol.6, No.5, , 2008.

[4] 畑田充弘, 他, “マルウェア対策のための研究用データセットとワークショップを通じた研究成果の共有, MWS2009”, 2009 年.

[5] D.Dagon, et.al, “A Taxonomy of Botnets”, In Proc. of NDSS’05, 2005.

[6] G.Gu, et al., “BotMiner: Clustering Analysis of Network Traffic for Protocol and Structure Independent Botnet Detection”, In Proc. of 17th USENIX Security Symposium, 2008.

[7] T.F.Yen et al., “Traffic Aggregation for Malware Detection”, In Proc. of DIMVA’08, 2008.

[8] A. Srivastava, et al., “Tamper-Resistant, Application-Aware Blocking of Malicious Network Flows”, In Proc. of RAID’08, 2008.

[9] B. D.Payne, et al., “Secure and Flexible Monitoring of Virtual Machines”, In Proc. of ACSAC’07, 2007.

[10] D.Dean, et al., “An Algebraic Approach to IP Traceback”, ACM Transactions on Information and System Security, Vol. 5, Issue 2, 2002.

[11] S.Stover, et al., “Analysis of the Storm and Nugache Trojans: P2P is here”, ;LOGIN, Vol. 22, No. 6, 2008.

[12] T.Holz, et al., “Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Strom Worm”, In Proc. of LEET’08, 2008.

[13] Xen Webpage, http://www.xen.org/, accessed at 08/06/2009.

[14] The Volatility Framework Webpage, http://www.volatilesysystems.com/default/volatility/, accessed at 08/06/2009.

[15] B. D.Payne, et al., “Lares: An Architecture for Secure Active Monitoring Using Virtualization”, In. Proc. of IEEE S&P, 2008.

[16] Virus.Win32.Virut.n, http://www.viruslist.com/en/viruses/encyclopedia?virusid=186407, accessed at 08/06/2009.