情報処理学会論文誌　Vol. 50　No. 9　2137–2146 (Sep. 2009)

通信の共通性を利用した悪性プログラム検知手法の実装と評価

水谷正慶[†1]　金井瑛[†1,*1]
武田圭史[†2]　村井純[†2]

本稿では未知の悪性プログラムの亜種を発見するために，悪性プログラムの共通した通信の特徴を利用して未知の悪性プログラムの亜種を高精度に検知する手法の実装，評価について述べる．悪性プログラムの一種であるボットは頻繁な亜種の発生や，一般にユーザによって利用されるネットワークプロトコルを模倣するなどによって監視を回避する傾向があり，正確に検知することが困難になっている．本稿では複数種のボットに共通する挙動が見られる点を利用し，ボットの活動モデルを用いて検知する手法を実装した．本実装はルールに基づき，複数の通信の相関関係を利用してボットの挙動を表現するものであり，多様なボットに対して柔軟に対応できる．本稿ではボットの通信と実ネットワークの通信データを用いて実装の精度を評価し，従来手法より高精度にボットを検知できることを示した．

A Malware Detection Method Based on Communication Commonality-implementation and Evaluation

Masayoshi Mizutani,[†1] Akira Kanai,[†1,*1]
Keiji Takeda[†2] and Jun Murai[†2]

This paper describes the implementation and the evaluation of a precise malware detection method based on communication commonality. Bots, which are a type of malwares, are difficult to be accurately detected because of their frequent updates and their ability to disguise malware communication as innocent ones. We have focused that there are communication commonality among many different types of bots, and we have implemented a detection method based on bots' activity models. The implementation is able to describe bot behaviors through correlations between communications, and it can flexibly respond to a huge variety of bots. In this paper, we have evaluated detection accuracy of the implementation using bots' communication data and real network traffic data, and showed that the implementation is more accurate compared to the existing methods.

1. はじめに

近年，コンピュータウィルスに代表される悪意あるプログラム（マルウェア）の活動が深刻化している．特にインターネット上の攻撃者が Command & Control（C&C）サーバを経由して任意のコマンドを多数の感染ホストに実行させるボットネットによる被害が問題となっている．ボットネットを構成するボットは C&C サーバから送信される命令に従い，分散型サービス妨害攻撃（DDoS）や迷惑メールの送信，個人情報の窃盗，他ホストへの感染を実行する[1)]．これに対し各ネットワークのセキュリティ管理者は，内部情報の漏洩阻止や対外的な信頼性維持のため，管理ネットワーク内のボットを検知，駆除する必要がある．しかし，ボットは C&C サーバからの指示によってボット自身のソフトウェアを更新する機能を持つため亜種の発生が頻繁になっており，ボットごとに固有に定義されるデータパターンなどを用いた検知手法では検知できないものが増加している．さらに，ボットは迷惑メール送信や DDoS の請負，あるいは機密情報の売買[2)] などによるビジネスモデルが確立しており，ボットの作成者もボットが検知，駆除されるのを防ぐために，目立ちにくい通信方法を実装する傾向がある．このため，特徴的な通信のパターンを検知条件とするネットワーク検知手法では十分な精度が期待できない．

本稿では通信の相関関係を利用し，未知のボットの活動を高精度に検知する手法を実装，評価した．侵入検知システム（IDS）は通信のヘッダおよびペイロードの検査によって，ネットワーク上で発生する様々なセキュリティ侵害に関するイベントを検知している．しかし，パケットや TCP セッションごとに検査しているため，複雑なイベントの検知が困難であった．これを解決するため，筆者らは TCP や UDP の送信元ポート，宛先ポートの組合せをセッションとして扱い，セッション間の相関関係を利用することで複雑なイベントを高精度に検知する手法[3)] を提案し，実装[4)] している．この手法は既知のボットの通信や P2P ファ

†1 慶應義塾大学大学院政策・メディア研究科
Graduate School of Media and Governance, Keio University
†2 慶應義塾大学環境情報学部
Faculty of Environment and Information Studies, Keio University
∗1 現在，NTT コミュニケーションズ
Presently with NTT Communications

イル交換ソフトウェアを高精度に検知するための手法として用いられることがあったが，未知のボットの検知にも応用できることが確認された．Gu らの研究[5] や筆者らの調査[6] ではボットの活動パターンにおける共通性が示されているが，既存の IDS では共通の活動モデルが分かったとしても，複雑な活動パターンを検知するための柔軟性に乏しく，これらの共通性を用いてボットを検知するには限界があった．本稿では筆者による実装[4] の機能を拡張し，柔軟性を備えた検知システムを実現した．さらに，本実装の有効性を示すために，実際のボットの通信と実運用ネットワークの通信を用いて検知の見逃しである False Negative（FN）と誤って検知してしまう False Positive（FP）の発生率を調査し，頻繁に発生するボット亜種への有効性を示した．

2. 既存手法によるボット検知の問題

セキュリティ管理者がネットワーク内のボットを検知する場合，ネットワークトラフィックを監視してボットの通信を発見するのが効果的である．これは，ボットが感染したホスト上で動作するセキュリティ対策ソフトの機能を阻害する点や，OS の機能を改ざんしてボットの活動を隠蔽する点などがあげられる．しかし，従来の不正な通信のパターンを検知するネットワークトラフィック監視手法を用いた IDS では，亜種の発生が頻繁であり，悪意ある活動の通信の巧妙化が進むボットの検知が困難となっている．

ボット作成者はボットの機能拡張やボットの検知，駆除防止のため，頻繁に亜種を作成し配布しており，2008 年 8 月のサイバークリーンセンターの調査[7] では収集したボットを含むマルウェア検体のうち 18%以上が市販のセキュリティ対策ソフトでは検知できなかったと報告されている．これは，現在の IDS やセキュリティ対策ソフトは検知精度を上げるために，個々のマルウェアの種類ごとにバイナリパターンを定義したシグネチャを作成しているためである．新しい亜種の出現からシグネチャを作成し検知可能になるまでの時間差が発生してしまうと考えられる．

さらに，セキュリティ対策ソフトでボットが検知できない場合が多くなる理由として，悪意ある通信の巧妙化が進んでいる点もあげられる．Paul らの調査[8] によれば，C&C サーバからのコマンドを受信するためのコントロールチャネルには Internet Relay Chat（IRC）が多く利用されており，通信内容にも特徴的な文字列を避ける傾向が見られる．既存手法でネットワーク通信からマルウェアを見つける場合は，マルウェア独自の通信プロトコルや通信内容に着目し，シグネチャを作成していたが，ボットが一般に用いられるプロトコルを利用したり，特徴的な文字列による通信を避けたりした場合，既存手法では誤検知が発生しやすくなる．

既存手法の検知精度を示すため，筆者らの環境において 2008 年 10 月 1 日から同月 31 日までに Nepenthes 0.2[9] で収集した 188 種類のマルウェア検体を実験環境下の VirtualBox 2.0[10] 上で Windows XP を動作させ，検体を実行し，検体実行後，それぞれ 1 時間ずつ発生するトラフィックを収集した．筆者らの調査[6] によってボットによるものと認められた 71 件のトラフィックデータを既存の IDS である Snort 2.8.1.2[11] によって検査した．検査には 2008 年 10 月 30 日版の Sourcefire VRT Certified Rules と Emerging Threats[12] で配布されているシグネチャを用いた．この結果，41 件のトラフィックデータからはボットの活動を特定し検知したが，残りの 30 件は検知できなかった．各通信からはなんらかのセキュリティ侵害の可能性を示すイベントとしては検知されたが，これらのイベントからボットの活動を特定，検知するためにはボットに関する知識に基づく追調査が必要であり，多くの管理者はこのような知識や経験などがないため，出力された情報をボット対策として有効に活用するのは困難であると考えられる．

3. 関 連 研 究

これまでに，通信の状態遷移を利用したシグネチャを利用して検知精度を高める手法としては NFR[13] や Netscreen-IDP の Stateful signature[14]，藤田の手法[15] などがあげられる．これらは個々のセッションごとの状態遷移と，各状態において検知に利用する不正な通信のパターンを定義できるため，通信に特徴が少ないボットを検知する場合でも複数の条件を組み合わせて設定可能となり，検知の精度が向上する．しかし，定義できるパターンは個々のセッションに限られるため，対応できるパターンに限界がある．

IDS などの監視装置が検知したイベントからシステムやネットワークの状態遷移を把握し，ある状態に到達した時点でセキュリティ侵害が発生したと見なす手法を，本稿ではイベント相関分析と呼ぶ．研究としては Zhou らの手法[16] や STATL[17]，NetSTAT[18] があげられる．これらの手法は ArcSight[19] や OSSIM[20] といったソフトウェアでも用いられている．イベント相関分析はイベントの発生順序や繰返しの回数によってセキュリティ侵害を検知する機能を備えている．しかし複数のイベントのうちいずれかが発生した場合を真とする条件（和条件）や，複数イベントのすべてが任意の順序で発生した場合を真とする条件（積条件）などの複雑な条件を指定できない．

BotHunter[5] は外部のボットによる攻撃から感染した内部ホストが，悪意ある活動を開始するまでをモデル化した検知手法である．様々な手法によって感染前から感染後に発生す

るイベントを検知し，状態遷移に利用している．状態遷移も複数の遷移パターンを想定しており，柔軟性があるといえる．ただし，検知のためにモデル化しているパターンが感染前から感染後への遷移に限られているため，外部から持ち込まれたホストがボットに感染していた場合の検知は困難だと考えられる．また，内部ホストが提供しているサービスの脆弱性を利用した攻撃を想定しており，電子メールの添付ファイルによる感染や Web ブラウザの脆弱性を利用した感染には対応が難しいと見られる．

また，ボットの通信傾向を統計的にモデル化し，検知する手法[21),22)]があげられ，それぞれ DNS の問合せに関するトラフィックの異常によって，ボットの検知を試みている．Binkley らの手法[23)]は TCP パケットの種類の割合を調べ，ボットらしい傾向を検知する．これらの検知手法はそれぞれのシナリオに適合する場合は有効に機能するが，モデルそのものは固定化されており，多様な活動モデルに対応できないという問題がある．また，統計的解析を用いた検知手法はネットワーク環境ごとに調整が必要になるため，運用には一定の条件が求められる．

4. 活動のモデルに基づいた未知のボット検知

4.1 アプローチ

本稿では複数のボットに共通する動作を利用することで，新たに出現した亜種に対しても有効な検知手法を提案する．多くのボットの亜種は細部が異なるが，おおまかな通信内容や動作手順などに共通する部分が見られる．たとえば，Paul らの調査[8)]では C&C サーバからの命令形態が示されているが，このような命令に関する処理は新しい亜種でも大幅に変更されない傾向がある．また Guofei らの研究[5)]や筆者らの調査[6)]でも，複数種類のボットに共通する動作がある点を指摘している．そのため，共通する動作をモデル化すれば新しいボットの亜種も既知のボット同様に検知ができると期待される．

図 1 では $E_{1,2,3,4,5}$ の 5 種類のイベントの組合せによってボットの活動を検知するモデルの例を示している．このモデルは C&C サーバとの通信とボットとしての活動の両方を発見した場合に，ボットを検知したと見なすモデルである．たとえば E_1 と E_2 がともに発生した場合（$E_1 \cap E_2$ と表記する）は IRC による C&C サーバとの通信を，E_3 が発生した場合は HTTP による C&C サーバとの通信をそれぞれ検知する．このどちらかを発見した場合（$(E_1 \cap E_2) \cup E_3$ と表記する），C&C サーバとの通信があったと判断する．さらに E_4 がボット更新用の Windows 実行形式のファイルダウンロード，E_5 が SMTP による頻繁なメール送信の試みを示している．この 2 つのどちらかが検知された場合（$E_4 \cup E_5$ と表記する），ボットに類する活動があったと判断する．このように活動モデルに基づいた検知では，各通信の細部を検知条件として設定することなくボットの亜種が検知できる．

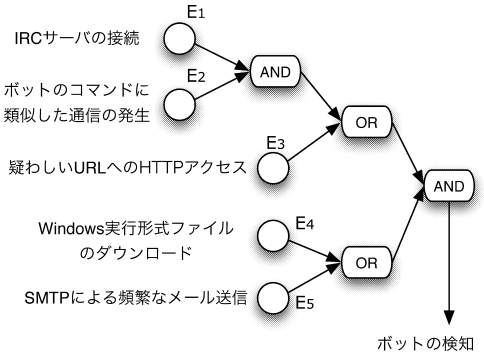

図 1　ボットの活動パターンのモデル例
Fig. 1　An example of bot activities model.

本アプローチはボットの活動を鳥瞰的にとらえた検知手法であるため，ボットの活動の細部が変更されることに対して耐性が高い．従来の IDS による検知では各ボットの個別のパターンを検知対象としているため，検知対象としている通信内容がわずかでも変更されると検知が回避されてしまうのに対し，本アプローチではボットの各動作を個別に検知するだけではなく，動作の全体像を把握することでボットを検知している．たとえば図 1 では，E_4 の通信が暗号化されても E_5 の動作が確認できればボットの活動として検知できる．あるいは $E_{1,2}$ のコマンド体系が大幅に変更されたとしても，E_3 を監視することで検知できる．このように本手法は頻繁に細部が変更されるボットに対して耐性が高く，ボット検知に対する有効性が高いと考えられる．

4.2 要 求 事 項

既存研究でも単純なモデルによる検知に応用できる手法や，モデルに基づいた検知を試みている手法があるが，以下で説明するように柔軟性と拡張性の点において限界がある．

(1) **検知モデルの柔軟性**

本稿では，より複雑な動作を表現できることを柔軟性と呼ぶ．3 章で述べたように，IDS の検知精度を高めるためにセッションの状態を管理する既存手法がある．しかし，これらの既存手法は検知対象が同一のセッションに限られるため，複数のセッションに分散したイベントを適切に処理できない．また既存のイベント相関分析手法では，

逐次的なイベントの遷移，またはイベントの繰返しを条件として扱えるが，図 1 で示した例のような複雑な積条件や和条件を処理できない．これらの手法は汎用的にボットを検知するにはモデルの柔軟性が不十分であり，より多様なイベントの関係を処理できる機能が必要となる．

(2) **検知モデルの拡張性**

本稿では多様なモデルが表現できることを拡張性と呼ぶ．BotHunter や統計的解析をもとにした手法ではボットのモデルを作成し，それに基づいて検知を試みている．しかし，既存の各検知手法は画一的なモデルのみで検知しているためモデルの追加・改善ができず，まったく新しいボットが発生した場合に対応が難しい．新しく発見されるボットの大部分は既知のボットの細部が変更された亜種だが，ごくまれに大幅に活動モデルが変更されたボットも出現している．そのためモデルを追加，修正できる拡張性はボット対策において重要であるといえる．

(3) **検知精度**

セキュリティ侵害を検知する手法は検知精度の高さが要求される．精度が低い検知手法は，検知結果が正しいかを確認するための追調査が必要となる．これは，セキュリティ管理者の負担を増加させる．さらに検知手法の信頼性が低下してしまう．検知精度は誤検知の少なさが指標となる．誤検知は検知するべきではない事象を検知してしまう FP と，検知するべき事象を見逃してしまう FN の 2 種類があり，この両方が低いほど検知精度が高いとする．

4.3　柔軟性と拡張性を考慮した検知手法の設計と実装

実装は，以前に筆者らが以前実装したネットワーク監視型のセキュリティ侵害検知機構である ROOK [4),24)] をもとに，複雑なボットの活動モデルを表現できるように拡張した．ROOK は異なるセッションの相関関係をルールによって記述し，セキュリティに関するイベントを検知する．ルール中にパラメータと呼ばれる変数が記述でき，このパラメータを利用してセッションの発生順序の検知や，通信に含まれるデータの一時的な保持，参照を実現している．

本稿での主な拡張の 1 つ目は，より複雑なセッションの発生順序を扱うための条件処理に関する拡張である．これは，図 1 で示した積条件や和条件を適切に処理するための拡張である．ボットの複雑な活動モデルを表現するためには，これらの条件への対応が必要となる．拡張前の実装では逐次的なセッションの発生順序しか表現できていなかったため，本実装では複数のパラメータを処理する機能を実装することで，複雑な条件への対応を実現した．パラメータは，あるイベントが発生した場合に指定された値を保持する書き込みと，イベント発生を判断する条件評価において保持している値を参照される読み込みの 2 つの機能がある．以前の実装では同時に扱えるパラメータが 1 つであったため，イベント E_a が発生した場合にパラメータ P_a に値を書き込み，E_b の発生条件を評価する際に P_a の値の変化を読み込むことで，E_a と E_b の発生順序を検知していた．この実装を改善し，複数のパラメータに対して同時に書き込み，読み込みができるように変更した．これによって E_a，E_b の逐次的な発生を検知するだけではなく，E_a と E_b のどちらかが発生したことを検知する $(E_a \cup E_b)$ や，E_a と E_b の両方が任意の順序で発生したことを検知する $(E_a \cap E_b)$ が表現できるようになった．さらにパラメータの値を書き込む際に，保持している値に対して加算，減算ができる機能を実装することにより，パラメータを 1 ずつ加算するような書き込みを可能とし，イベントの繰返しを処理している．

2 つ目の主な拡張は，各セッションにおける状態遷移処理の実装である．たとえば図 1 での E_3（疑わしい URL への HTTP アクセス）では，HTTP の要求に含まれる URL とアクセスに対する成否の両者を適切に検知する必要がある．ボットは C&C サーバからの命令の受信によって活動内容を決定する場合が多いため，アクセスの成否は検知にとって重要な手がかりとなり，各セッションの状態遷移を管理する機能が求められる．これは文献 13)–15) において実装されている機能を包含するものであり，各セッションに任意の数の状態を作成し，各状態遷移の条件を設定できる．さらに，ある条件と通信内容が一致した場合，もしくはある状態へ遷移した場合に指定された動作を実行する機能も付与した．これによって各プロトコルの要求と応答やコマンドの送受信状態などを適切に検査するルールが記述できる．

4.4　本実装によるセッションの相関関係処理の流れ

図 2 は本実装におけるセッションとパラメータの関係と処理の流れを例として示している．図ではセッション 1 のイベント（E_1）とセッション 2 のイベント（E_2）が発生した後にセッション 3 のイベント（E_3）が発生したことを検知するルールである．4.1 節の表記に従うと $(E_1 \cap E_2)E_3$ と表せる．この表記では E_1 と E_2 の発生順序は任意であり，E_3 は $(E_1 \cap E_2)$ の後に発生した場合のみ検知される．

本実装は各セッションでの状態管理と，1 つ以上のパラメータを用いたセッションの発生順序に基づいた検知を実現している．図中では 3 つのセッションと 2 つのパラメータによって $(E_1 \cap E_2)E_3$ が処理されている．セッション 1，2，3 にはそれぞれ $S_{1,2}$，$S_{3,4,5}$，$S_{6,7}$ の状態が存在する．各状態には遷移条件（$T_{1,2,3,4,5}$）があり，監視している通信内容と条件が一致した場合に状態を遷移する．図中ではセッション 1，2 の S_2 と S_5 がそれぞれ E_1，E_2

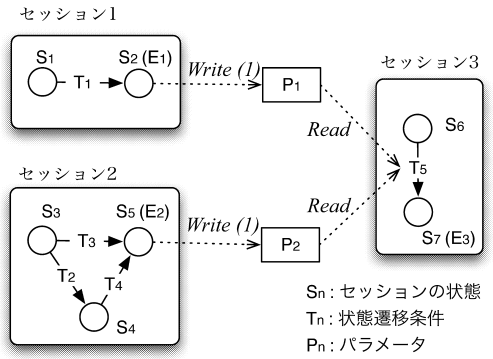

図 2　セッションの状態遷移とパラメータの構造
Fig. 2　An architecture of session state transition and parameter.

に対応しており，各状態に到達した場合に検知条件に利用されるパラメータが操作される．図中ではパラメータ P_1，もしくは P_2 に対して “1” を書き込む．そして，T_5 は条件の成立を検査する際，P_1 と P_2 の値を同時に検査しており，両者の値が 1 であることを確認する．これによって積条件である $(E_1 \cap E_2)$ の成立を検査する．

5. 評　　価

5.1　検知精度評価の概要

活動パターンをモデル化することによってボットの存在を通信データから検知する本手法の有効性を示すために，その検知精度を評価した．評価ではボットの活動パターンを示す 3 種類のルールを作成し，ボット通信の見逃しである FN と誤検知である FP の発生率をそれぞれ調査した．

また，比較対象として Snort [11] も同様のデータで評価し，検知率を比較した．Snort は最も利用されているシグネチャ型 IDS の 1 つであり，シグネチャも頻繁に更新されている．またシグネチャの種類も豊富であり，感染前の攻撃や感染後の活動などの様々な状況を検知するためのルールが用意されているため，比較対象に選択した．

5.2　検知精度評価用のルール

本手法の検知精度を評価するために，IRC によって命令を送受信するボットの主要な活動パターンを示すルールを作成した．これを**表 1** に示す．ルール R_1，R_2，R_3 はそれぞれイベント $E_{1,2,3,4,5}$，$E_{6,7,8,9}$，$E_{10,11,12}$ で構成されている．イベントの発生条件を満たすと指定された動作を実行し，最終的にボットの活動を検知した場合は管理者に対応を促すアラートが発行される．各イベントの動作ではパラメータ P_n にあらかじめ指定した値や通信データの一部を代入し，発生順序の制御や通信データの保持を実現している．

表 1　評価用ルール詳細
Table 1　Rules for evaluations.

		イベントの発生条件	イベント発生時の動作
R_1	E_1	TCP セッションにおける IRC の開始	$s \leftarrow S_1$
	E_2	$(s = S_1) \cap$ TCP 通信において"135", "137", "139", "445", "msass", "root.mass", "ipscan", "advscan", "adv.start", "lsass", "scanall"のいずれかを含む文字列が出現	$P_1 \leftarrow true$
	E_3	$(s = S_1) \cap$ Base64 形式で 30 字以上の文字列を含む通信の発生	$P_1 \leftarrow true$
	E_4	$(P_1 = true) \cap$ TCP ポート 135，137，139，445 番への接続要求	$P_2 \leftarrow P_2 + 1$
	E_5	$P_2 > 32$	アラート
R_2	E_6	TCP セッションにおける IRC の開始	$s \leftarrow S_2$
	E_7	$(s = S_2) \cap$ URL 形式の文字列の出現	$P_3 \leftarrow$URL 形式の文字列
	E_8	$P_3 =$HTTP の取得要求における URL	$s \leftarrow S_3$
	E_9	$(s = S_3) \cap$ HTTP 応答が Windows 実行形式のファイル	アラート
R_3	E_{10}	TCP ポート 135，137，139，445，3127 番への接続要求	$P_4 \leftarrow P_4 + 1$ $P_5 \leftarrow$ 送信元 IP アドレス
	E_{11}	$(P_5 =$ 送信元 IP アドレス $) \cap$ Windows 実行形式のファイルの送信	$P_6 \leftarrow true$
	E_{12}	$(P_4 > 32) \cap (P_6 = true)$	アラート

* s は検査対象のセッションの状態とする．

R_1 は IRC で受信した C&C サーバからの命令に基づいて，TCP ポート 135，137，139，445 のいずれかを調査する活動を検知するルールである．IRC 上での命令に使われる文字列は Cooke らの研究[25] や Cyber-TA による調査[26] において明らかにされており，E_2 ではその一部を用いている．また，Base64 形式の命令と見られる文字列も E_3 で検知対象としている．IRC セッションに命令と予想される文字列が発見された後に各 TCP ポートへの接続要求が一定回数を上回った場合，ボットの活動と見なす．

R_2 は IRC で受信したボット更新ファイルの取得命令に基づいて，HTTP 経由で実行ファイルを取得する振舞いを検知するルールである．E_7 において IRC 上で発見した URL を P_5 に保存し，E_8 の HTTP 要求と P_5 の URL が一致するかを確認する．E_9 は E_8 からの状態遷移であるため，E_8 と同一セッションでの HTTP 応答に Microsoft Windows において実行可能な形式のファイルが含まれていた場合，これをボットの活動と見なす．実行形式の検知は文献 27) を参考にルールを作成した．

R_3 は同一ホストから調査活動と Windows 実行形式のファイル送信が同時期に発生していることを検知するルールである．TCP ポート 135，137，139，445 を用いた調査活動を検知し，同時に送信元アドレスを P_7 に保存する．当該ホストから送信されるパケットに Windows 形式を検知した場合，ボットの感染活動および悪性プログラムの送信を実施していると見なす．

比較対象の Snort は公式サイト[11] で配布されている Sourcefire VRT Certified Rules と Emergin Threats[12] で配布されている 2008 年 11 月 27 日版のルールを利用した．具体的にはボットを含むマルウェアの活動を検知するための emerging-virus.rules，emerging-malware.rules，spyware-put.rules，virus.rules，backdoor.rules から 3,387 件のルールを利用した．

5.3　悪性プログラムの通信データを用いた FN 発生率調査

FN の発生率を確認するために，筆者らがハニーポットから収集した悪性プログラムを利用した．ハニーポットは 2008 年 6 月から同年 10 月にかけて Nepenthes[9] を運用し，936 件の悪性プログラムを収集している．Nepenthes は攻撃コードを受信した場合に，攻撃対象となったソフトウェアの脆弱性を模倣してプログラムを取得する．よって，ハニーポットによって収集されたプログラムは悪性であると見なせるため，これらを実行した際に発生した通信には検知するべきデータが含まれているといえる．本評価では悪性プログラムを自動的に実行して得られた通信データに対して，本手法が活動の見逃しが発生しないかを調査した．

ハニーポットによって収集した悪性プログラムを 2008 年 11 月 1 日から同月 6 日の間に実験環境で実行し，FN 評価用の通信データを取得した．実験環境は 2 章と同様に VirtualBox 2.0[10] 上で修正パッチを適用していない Windows XP Professional をゲスト OS として動作させた．ゲスト OS は Network Address Port Translation（NAPT）を用いて構成されたネットワークに接続しており，外部からの接続要求はできない．また，外部への被害防止のため外向きの TCP/UDP ポート 25，135，137，139，445，1433，1434 をそれぞれ遮断し，転送速度を制限した．収集された悪性プログラムは起動直後のゲスト OS 上で実行し，その後 1 時間かけて通信データを収集した．ボットは C&C サーバからの命令に応じて悪意ある活動を実行したり，ボット自身のソフトウェアを更新したりするため，活動パターンが変化して検知が難しくなる．そのため C&C サーバからの命令が機能していない場合はボットとしての脅威は低減されるため，通信が発生しないプログラムや，調査活動以外の活動が観測されなかったプログラム，コントロールチャネルから命令を受信しない，あるいは命令を無視するプログラムは除外し，ボットとしての活動が見られた 299 件の通信データを対象に検知性能を調査した．

表 2　ボット検体による通信データを用いた FN 評価結果
Table 2　A Result of FN evaluation.

	Snort で検知成功	Snort で検知失敗	合計
本手法で検知成功	219	75	294 (98.33%)
本手法で検知失敗	2	3	5 (1.67%)
合計	221 (73.91%)	78 (26.09%)	299

* P_n: パラメータ
* 割合は小数点第 3 位以下を四捨五入

表 2 は FN の評価結果を示している．それぞれ本手法による検知の成功，失敗と Snort による検知の成功，失敗の内訳を示している．本手法は検知すべき 299 件の通信データに対して，全体の 98.33%である 294 件を検知している．しかし，Snort は全体の 73.91%である 221 件しか検知できていない．さらに，Snort で検知できなかった 78 件中，75 件は本手法で検知している．これは Snort と比較して本手法の見逃しの発生率が大幅に少ないことを示している．一方で本手法が見逃してしまった 5 件のうち，2 件は Snort で検知している．さらに，どちらの手法でも検知できなかった通信データは 3 件であった．

5.4　FN 評価の結果に対する追調査

FN 評価において本手法が検知できなかった 5 件に対し，追調査を実施した．検知できなかった通信データについて調査したところ，この 5 件は主に HTTP 通信をベースにした活動モデルであり，IRC によるコントロールチャネルは持たなかった．本稿の評価で用いた $R_{1,2}$ は IRC によるコントロールチャネルを想定したモデルであり，HTTP による通信を基盤としたボットのモデルは含まれていない．しかし，この 5 件の活動パターンを分類したところ 3 件には共通性が見られたため，モデル化が可能である．また，残りの 2 件もそれぞれモデル化に利用できると考えられる活動パターンの特徴がいくつか見られた．よって，HTTP 通信による検知用モデルを作成することで，これら 5 件の検知が可能になると考えられる．

5.5　実運用ネットワークの通信データを用いた FP 発生率の調査

FP の発生率を評価には 3 種類の実運用ネットワークの通信データを用いた．意図的にボットの通信を発生させている 5.3 節の通信データとは異なり，通信データに含まれるボット通信の件数を正確に把握するのは困難である．そのため，実運用ネットワークの通信データは FP の発生率調査のみに利用し，検知数は評価の対象としない．

表 3　FP 評価用通信データ概要
Table 3　Traffic data set overview for evaluations.

	収集開始日時	期間	パケット数	データ長	説明	内部ホスト数
D_1	2008-11-04 12:00	14 時間	134.45 M	69.26 GB	研究ネットワーク	約 500 台
D_2	2008-09-10-14:00	26 時間	1139.16 M	48.06 GB	カンファレンスのネットワーク	約 250 台
D_3	2008-11-05 13:00	5 時間	743.10 M	513.97 GB	バックボーンネットワーク	約 100,000 台

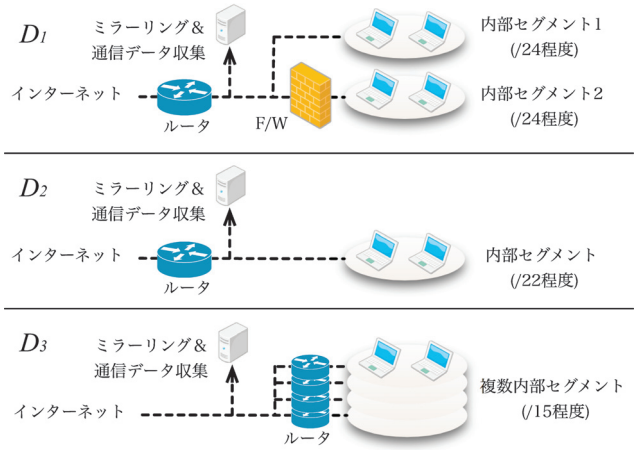

図 3　FP 評価用通信データ取得環境概要
Fig. 3　An overview of traffic capturing environments to evaluate FP.

表 3 は各通信データの概要を示しており，**図 3** は各通信データの収集環境を示している．D_1 は筆者らが定常的に運用している研究ネットワークの通信である．ファイヤウォールによって外部からの接続が遮断されているセグメントと，実験用に通信を制限していないセグメントが存在している．D_2 は 4 日間開催されたカンファレンスのネットワークである．特にファイヤウォールなどによる通信制限はない．D_3 はバックボーンネットワークであり，複数組織が接続しているネットワークである．各組織の運用ポリシは様々であり，観測点は各組織内ルータの上位となっている．また各ネットワークにおいて Web サーバ，メールサーバ，DNS サーバなどのサービスホストは各セグメント内部に設置されている．

検知には 5.3 節の調査同様，本手法による $R_{1,2,3}$ および Snort を利用した．ただし，検知結果が実際のボットの活動を検知したのか FP なのかを判断するために，Snort で利用したルールは 5.3 節で検知されたもののみを利用した．

表 4　FP 評価結果
Table 4　A result of FP evaluation.

手法	ルール	D_1	D_2	D_3
本手法	R_1	0	0	0
	R_2	0	0	0
	R_3	17 (2,450)	0	7 (942)
Snort	BACKDOOR DoomJuice/mydoom.a backdoor upload/execute attempt	3 (7)	0	1 (1)
ボットの活動が確認された IP アドレス数		20	0	8

* すべての通信データにおいて検知数が 0 だった Snort のルール，および明示的にボットの活動を示しておらず誤検知の可能性が高いルールは表から除外している．
* 数値が通信データ中でボットとして検知された IP アドレス数，括弧内の数字が各種法での検知を示す．

表 4 に FP 発生率調査の結果を示す．表は各データセットに対し，本手法がボットの活動を検知したルールと Snort が検知したルールの各検知数を示している．ただし，パケットを検知対象とする Snort と複数のセッションを検知対象とする本手法とでは検知数の数え方が異なるため，括弧内に一意な攻撃元 IP アドレスの数を示している．本手法が検知した R_3 は，ボットによる調査活動および攻略済みホストに対する Windows 実行形式ファイルの送信を検知するルールである．D_1 で発生した 2,450 件と D_3 で発生した 942 件を調査したところ，D_1 では 17 個，D_3 では 7 個の外部の IP アドレスから筆者らが運用するハニーポットに対して攻撃している通信であった．これはハニーポットのログからも同時間帯に攻撃があり，Windows の実行形式ファイルがハニーポットホストに送信されていたことを確認している．よって，これらの検知結果は FP ではなくボットの活動であると判断できる．一方，Snort が検知している “BACKDOOR DoomJuice/mydoom.a backdoor upload/execute attempt” はすでにボットに感染しているホストに対して Windows 実行形式のファイルを送信する際のプロトコルを検知している．D_1 で発生している 7 件と D_3 で発生している 1 件も本手法の検知と同様にハニーポットへの攻撃であることを確認した．これらの結果から，両手法はともに 5.3 節でボットを検知したルールでは FP が発生していないことが明らかになった．

また，本手法と Snort の検知結果の内容を比較したところ，Snort と本手法が別のイベントを検知していることが明らかになった．Snort で検知していたのは外部から内部への攻撃

で，調査活動をせずに 1 ホストに対する攻撃を検知していた．筆者らが運用しているハニーポットは検体収集を目的とした Nepenthes であり，攻撃受信後はボットの活動を一部のみ模倣する．そのため，本手法では感染したホストとして検知しなかった．一方，本手法で検知したイベントに対して追調査を実施したところ，5.3 節とは異なる Snort のシグネチャによってボットの活動を検知した．

5.6　拡張性と柔軟性に着目した既存手法との比較

本手法は XML によって記述されたルールによって動作しているため，容易にモデルの追加，変更ができ，拡張性に優れているといえる．既存手法では BotHunter[5] や Binkley らの手法[23]，朝長らの手法[22] が悪性プログラム特有の活動モデルを利用した検知手法としてあげられるが，これらの手法は固定されたモデルを採用しているため，容易なモデルの追加，変更が難しいという欠点がある．短期的には一定のモデルでボットが検知できたとしても，長期的には活動パターンが大幅に変更されたボットが出現すると予想されるため，本手法はモデルの拡張性から長期的なボット対策として有効であると考えられる．

また，本手法は筆者らによる以前の実装[4] と比較し，イベントの積条件，和条件，繰返しを検知条件として実装することで柔軟性を実現している．既存のイベント相関分析手法や従来の実装では，逐次的なイベントの発生順序や発生回数しか処理できない．よって，表 1 における R_3 のように，任意の順序で発生するイベントを利用してボットの活動を検知できない可能性がある．従来のイベント相関分析手法や筆者らの以前の手法で R_3 を表現する場合，E_{10} が 32 回発生した後に E_{11} が発生する条件と，E_{11} が発生した後に E_{10} が 32 回発生する条件のいずれかしか表現できない．そのため，E_{10} が 32 回に到達する途中で E_{11} が発生した場合，見逃しとなってしまう．本手法では E_{11} の発生と E_{10} の発生を個別に処理できるため，それぞれが任意のタイミングで発生しても見逃しが起きない．このような非同期的に発生するイベントを適切に処理できるため，柔軟性の点でも既存手法より優れているといえる．

6.　考察と今後の課題

5 章の評価において，本手法は従来のシグネチャ型 IDS の代表的な実装である Snort と比較して FN 発生率が低く，FP の発生もないことから，従来の手法と比較してボットの検知精度が高いと考えられる．評価に用いた配布されている Snort のシグネチャは Sourcefire VRT Certified Rules が 1 週間に約 1 回，Emerging Threats が毎日更新されており，評価時点での網羅性において最新の状態にあったといえる．また，ボットの収集期間が 2008 年 6 月から同年 10 月であり，通信データの収集が同年 11 月 1 日から同月 6 日までの間であるのに対し，本評価で使用したルールは 2008 年 11 月 27 日版となっている．よって，少なくとも 20 日以上が経過した時点でのルールを使用しており，実際のネットワーク監視運用と同様の条件であるといえる．これらの事実からシグネチャ型の IDS では検知に限界がある点と，本手法が亜種に対しても有効である点が明らかになった．

さらに，抽象的かつ拡張可能なルールを用いて検知できる点も有用性が高いといえる．本稿での評価において，本手法は 3 種類のルールを使用したのに対し，Snort は 3,387 件のルールを使用している．記述方法の違いや検知対象の違いがあるため単純なルール数の比較は難しいが，シグネチャ型 IDS やウィルス対策ソフトはシグネチャの肥大化が問題となっている．本手法では限られたルールで複数種類のボットを検知できるため，頻繁なルール更新の必要性が低いという点で優れている．ただし，本手法は作成されたモデルによっては具体的な攻撃や活動の内容を把握しにくいため，シグネチャ型 IDS と併用することでネットワークフォレンジックなどの調査が容易になると考えられる．

本実装は活動パターンを XML によって記述しているため，モデルの追加，更新，削除の負担が少なく，拡張性が高い．本評価で用いたモデルは同系統のボット亜種には有効だが，まったく新しい通信方式を採用したボットには対応できない．しかし，モデルの作成や拡張が容易になっているため，モデルが明らかになれば早急な対応が可能だといえる．

本手法も今回の評価で用いたモデルのみでは一部のボットの活動を検知できなかったため，さらなるデータ収集とモデル作成が必要となる．5.4 節で示したとおり，今回のモデルで検知できなかったボットのモデルを作成するのは可能だが，他のボットとの共通性を見極めるためにはより多くのボットを調査しなければならない．今後，IRC 以外のコントロールチャネルや P2P によるボットネットの形成が拡大する可能性は高く，より多くの種類のボット収集とモデル作成が必要であると考える．

7.　ま　と　め

本稿では高頻度で亜種が発生するボットに共通した通信がある点に着目し，ボット活動のモデル化を利用した検知手法を実装，評価した結果について述べた．頻繁に亜種が発生するボットに対し，通信の共通性を利用することで未知のボット亜種の検知が可能になることが見込まれ，これを実現するために，複雑な活動パターンをモデルとして表現できる柔軟性とモデルの拡張性を備えたボット検知手法を実装した．この実装によってボットの通信を検査したところ，299 件のボットの通信データのうち 98.33%をボットの活動として検知できた．

また，実運用ネットワークの通信データを検査したところ誤検知は発生しないことが確認された．以上の点から本手法がボット検知において有効であり，社会的な問題となっているボットへの対抗策になると期待される．

謝辞　本研究は日本学術振興会の助成を受けたものであり，日本学術振興会に感謝する．

参 考 文 献

1) Bacher, P., Holz, T., Kotter, M. and Wicherski, G.: Know your Enemy: Tracking Botnets (2005). http://www.honeynet.org/papers/bots/
2) Friess, N. and Aycock, J.: *Black Market Botnets*, MIT Spam Conference (2008).
3) 水谷正慶，白畑　真，南　政樹，村井　純：複数セッションの相関関係を利用したセキュリティイベント検知手法の提案，IPSJ CSEC コンピュータセキュリティシンポジウム 2007 論文集 (2007).
4) Mizutani, M., Shirahata, S., Minami, M. and Murai, J.: ROOK: Multi-session Based Network Security Event Detector, *SAINT*, pp.48–54 (2008).
5) Gu, G., Porras, P., Yegneswaran, V., Fong, M. and Lee, W.: BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation, *Usenix Security Symposium 2007* (2007).
6) 水谷正慶，武田圭史，村井　純：通信の状態遷移に着目したボット活動の調査，マルウェア対策研究人材育成ワークショップ 2008 (2008).
7) Cyber Clean Center: 2008 年 08 月度サイバークリーンセンター活動実績 (2008). https://www.ccc.go.jp/report/200808/0808monthly.html
8) Paul, B. and Vinod, Y.: An Inside Look at Botnets, *Special Workshop on Malware Detection, Advances in Information Security*, Springer Verlag (2006).
9) Nepenthes – finest collection. http://nepenthes.mwcollect.org/
10) Sun Microsystems, Inc.: VirtualBox. http://www.virtualbox.org/
11) Martin Roesch: Snort (1998). http://www.snort.org
12) Matt Jonkman: Emerging Threats (2008). http://www.emergingthreats.net/
13) NFR Security, Inc.: Network Flight Recorder. http://www.nfr.net
14) Juniper Networks Inc.: Stateful Signature, Context-based Packet Signature (Netscreen IDP). http://www.juniper.net/products/intrusion/detection.html#Stateful_Sign_Det
15) 藤田直行：侵入検知ポリシの記述性向上によりログ出力量の低減を可能とした不正アクセス処理システムの開発，電子情報通信学会論文誌，Vol.J88-D-1, pp.391–400 (2005).
16) Zhou, J., Heckman, M., Reynolds, B., Carlson, A. and Bishop, M.: Modeling network intrusion detection alerts for correlation., *ACM Trans. Inf. Syst. Secur.*, Vol.10, No.1 (2007).
17) Eckmann, S., Vigna, G. and Kemmerer, R.: STATL: An Attack Language for State-based Intrusion Detection (2000). http://citeseer.ist.psu.edu/eckmann00statl.html
18) Vigna, G. and Kemmerer, R.A.: NetSTAT: A Network-based Intrusion Detection Approach, *ACSAC* (1998).
19) ArcSight: Using Advanced Event Correlation to Improve Enterprise Security, Compliance and Business Posture, White Paper (2007).
20) Karg, D.: *OSSIM Correlation engine explained, Sample scenario: NETBIOS DCERPC ISystemActivator*, OSSIM (2004). http://www.ossim.net/docs/correlation_engine_explained_rpc_dcom_example.pdf
21) Musashi, Y., Ludena, R., Dennis, A., Nagatomi, H., Matsuba, R. and Sugitani, K.: A DNS-based Countermeasure Technology for Bot Worm-infected PC terminals in the Campus Network, *Journal for Academic Computing and Networking*, Vol.10, No.1, pp.39–46 (2006).
22) 朝長秀誠，田中英彦：Botnet の命令サーバドメインネームを用いた Bot 感染検出方法，情報処理学会研究報告，CSEC［コンピュータセキュリティ］，Vol.2006, No.129, pp.13–18 (2006).
23) Binkley, J.R. and Singh, S.: An Algorithm for Anomaly-based Botnet Detection, *SRUTI '06*, pp.43–48 (2006).
24) MIZUTANI, M.: ROOK Multi-session Baesd Network Security Event Detector (2008). http://rook.sourceforge.net/
25) Cooke, E., Bailey, M., Jahanian, F. and Mortier, R.: The Dark Oracle: Perspective-Aware Unused and Unreachable Address Discovery, *The 3rd ACM/USENIX Symposium on Networked Systems Design and Implementation* (2006).
26) Cyber-TA Research and Development Project: SRI Honeynet and BotHunter Malware Analysis Automatic Summary Analysis Table (2005).
27) Visser, E.: PC Executable Format, Program-Transformation.Org: The Program Transformation Wiki (2000). http://www.program-transformation.org/Transform/PcExeFormat

(平成 20 年 12 月 1 日受付)
(平成 21 年　6 月 4 日採録)

水谷　正慶（学生会員）

修士（政策・メディア）．1983 年生．2006 年慶應義塾大学環境情報学部卒業．2008 年同大学院政策・メディア研究科修了．現在，慶應義塾大学大学院政策・メディア研究科後期博士課程在学中．日本学術振興会特別研究員 DC1 悪性プログラム対策や侵入検知システムを中心としたネットワークセキュリティの研究に従事．

金井　　瑛

修士（政策・メディア）．2007 年慶應義塾大学環境情報学部卒業．2009 年同大学院政策・メディア研究科修了．現在，NTT コミュニケーションズ勤務．

武田　圭史（正会員）

博士（政策・メディア）．2001 年慶應義塾大学大学院政策・メディア研究科後期博士課程修了．2002 年アクセンチュア株式会社勤務．2004 年カーネギーメロン大学情報ネットワーク研究所客員教員．2005 年カーネギーメロン大学大学院情報セキュリティ研究科（日本校）教授．現在，慶應義塾大学環境情報学部教授．

村井　　純（正会員）

博士（工学）．1979 年慶應義塾大学工学部数理工学科卒業．1981 年同大学院理工学研究科修士課程数理工学専攻修了．1987 年 1 月博士（工学）．現在，同大学環境情報学部教授．